Graphics Display System

Model 3404

System Reference Manual

Interactive Graphics Products

Graphics Display System

Model 3404

System Reference Manual

OCTOBER, 1977

VECTOR GENERAL INC., 21300 Oxnard Street, Woodland Hills, California 91364

Copyright (C) 1976 by Vector General, Incorporated

NOTICE

M110700REF

SYSTEM REFERENCE MANUAL

GRAPHICS DISPLAY SYSTEM, MODEL 3404

CONTENTS

CONTENTS (Continued)

ii

CONTENTS (Continued)

CONTENTS (Continued)

CONTENTS (Continued)

ILLUSTRATIONS

CONTENTS (Continued)

ILLUSTRATIONS (Continued)

CONTENTS (Continued)

TABLES

Frontispiece

SYSTEM REFERENCE MANUAL

GRAPHICS DISPLAY SYSTEM, MODEL 3404

SECTION I

GENERAL DESCRIPTION

1-1. INTRODUCTION

This manual describes the Model 3404 Graphics Display System* (see Frontispiece), manufactured by Vector General, Incorporated, 21300 Oxnard Street, Woodland Hills, California. Material contained in this reference manual is organized as follows:

Section I	General Description: scope, references, physical and performance characteristics, and vector drawing rates.
Section II	Functional Description: discussion of block diagram for the Model 3404 Graphics Display System.
Section III	Graphic Processor Programming: GPU objects, control, argument addressing, registers, element generation and USER INSTRUCTIONS.
Section IV	Graphic System Programming: enable and halt usage, display object, sub-objects, nesting for transformations, data scale, viewing parameters, windowing, picture transformations, three-dimensional viewing parameters, interaction-aid features (SELECT, HIT and EDIT), addressing, object list contents, stack operations, and object level transformations.
Section V	Display Controller Hardware Description: display registers, control registers, data registers and character draws.
Section VI	Glossary of Terms.
Appendices	Reference Material
Index	

*U.S. PATENT No. 3772563; other PATENTS PENDING

The Model 3404 is a fourth-generation, high-speed graphics system which retains the outstanding features of the Series 3 Graphics Systems and augments them with the following attractive improvements:

- Incorporation of a digital microprocessor, providing the facility for off-line computations to generate high-speed transformations, windowing, dynamic zooming and clipping, perspective, small element discard, circle/arc, cubic curve synthesis, and 3-dimensional depth cueing.

- Programming tasks have been significantly reduced through use of a powerful set of 46 user instructions. The microprocessor firmware permits a remarkable reduction in software coding requirements compared with systems of previous generations.

- User Interaction Aids have been supplemented with Select, Hit/Pick, and Edit features. In addition, a Writable Control Store feature permits the user to modify, extend, or replace entirely the microprocessor firmware.

1-2.   SCOPE

Information contained in this manual is intended for software personnel who must generate user programs for graphic displays in the Model 3404. In addition, material presented here can be extremely useful for engineers and supervisory personnel who desire an overview of the Model 3404.

1-3.   REFERENCES

Several reference manuals are available that complement material contained in this publication. Table 1-1 lists the most important references associated with the Model 3404 Graphics Display System. Volume I provides hardware data concerning the Refresh Buffer Unit, Display Control Unit, Vector Generator Unit, Font Generator Unit and Monitor Control Unit. Volume II describes hardware features of the Graphic Processor Unit. Volume III discusses firmware features of the Graphic Processor Unit for programmers. Volume IV, Writable Control Store, and Volume V, Maintenance Panel, are two hardware manuals concerned with the Model 3405 configuration. Finally, the Model 3404 Programming Concepts Manual provides information relating to user programs.

Table 1-1. Reference Manuals

Document Number	Title
M110700	Volume I. Series 3400 Graphics Display System
M110380	Volume II. Graphic Processor Unit Hardware
M110721-XXX (February 1978)	Volume III. Graphic Processor Unit Firmware
(April 1978)	Volume IV. Writable Control Store
(May 1978)	Volume V. Maintenance Panel
M113489	Model 3404 Programming Concepts Manual

1-4.   PHYSICAL DESCRIPTION

The Model 3404 is available in either self-contained console-type work stations or in standard rack-mountable configurations.   The standard 3404 work station consists of an attractive table which provides ample space to accommodate a table-top display monitor and all interactive devices associated with the station.   The display generator cabinet which houses the system electronics may either be positioned under the table or may be rolled out to provide additional working space if desired.   Tables 1-2 through 1-5 list the installation requirements and performance characteristics.

NOTE: The values specified in Tables 1-2 and 1-3 will vary with the system configuration selected by the customer.   Please contact your local Vector General representative for deviations from these values.

Table 1-2. Physical Characteristics

Item	Value
Display Monitor	
Height	57.15 cm (22.5 inches)
Width	68.58 cm (27 inches)
Depth	93.98 cm (37 inches)
Weight	58.97 kg (130 pounds)
Display Generator Cabinet	
Height	72.39 cm (28.5 inches)
Width	48.26 cm (19 inches)
Depth	78.74 cm (31 inches)
Weight	136.08 kg (300 pounds)
Table	
Height	73.66 cm (29 inches)
Width	152.40 cm (60 inches)
Depth	106.68 cm (42 inches)
Weight	68.2 kg (150 pounds)

Table 1-3. Power Requirements

Item	Value
Primary power (all models) AC connector: NEMA type 5-20P	105-125 Vac, 47-440 Hz, single phase or 210-250 Vac, 50 Hz, single phase
Consumption*	
Models 3402, 3403, 3404	1350 VA*
Model 3405	1580 VA*
* Assuming an FGU and DM21 monitor are included. Compute power consumption separately for optional configuration and devices	* Includes DM21 consumption of 550 VA average.

Table 1-4. Environmental Requirements

Item	Value
Temperature range	0 to 50° (32° to 122°F), operating -40°C to 71°C (-40°F to 160°F), non-operating
Relative Humidity	up to 95% at 40°C (104°), non-condensating
Altitude	up to 3038 meters (10,000 feet) at 25° operating temperature up to 7620 meters (25,000 feet), non-operating

Table 1-5. Performance Characteristics

Item	Value
GRAPHIC PROCESSOR	A high-speed graphic processor (GPU) and an associated refresh buffer (RBU) which converts raw data received in the display list from the host computer into a usable display refresh list for CRT refresh.
User instruction set	46 instructions, refer to Appendix A1.
Processing time*	26.13μs average to process 2-D line element (3404) 29.88μs average to process 3-D line element (3404) 54.75μs average to process clipped 2-D line element.(3404) 57.86μs average to process clipped 3-D line element.(3404)
RAM stack size	72 by 16, standard. Provides several levels of sub-object nesting in the graphics RAM stack. When the graphics stack is full, any amount of CPU memory may be used as a stack extension.
Windowing	A firmware feature in which a portion of display data within a designated volume may be extracted and presented for viewing.

*These values were obtained using a DEC PDP11 Interface to a PDP11/05 with no peripheral contention. In addition, the following conditions were valid:
    (a) GPU cycle time = 250μs
    (b) Lines instruction was used with these attributes:
        (1) Lines list: Immediate with terminate
        (2) Data format: Full word
        (3) Coordinates: Absolute

Table 1-5. Performance Characteristics (Continued)

Item	Value
GRAPHIC PROCESSOR (Continued)	
Clipping	The process by which data extracted for viewing in the window volume are limited to extraction boundaries specified by windowing parameters, viewporting boundaries, and display screen boundaries. This is accomplished by clipping and discarding unseen elements and recomputing endpoints of partially visible elements.
Small element discard	A feature in which the output of successive display elements is suppressed. This occurs when beam movement between these elements is virtually non-existent.
Viewporting	A firmware feature in which a portion of display screen is designated for presentation of display data within a specified window under chosen viewing parameters.
Zooming	Effect observed when window-extracted display data are magnified or reduced by changing relative size of window with respect to viewport.
3-D Depth Cueing	Feature attained through use of intensity modulation or perspective for enhancement of depth visualization
Edit aid	A hardware/firmware feature to provide tentative means of interactive deletion, insertion, or adjustment of displayed elements.
Direct Addressing	Hardware feature that provides means of accessing data (i.e., coordinate endpoints, rotation angles, etc.) from addressed locations.
Reference Addressing	Hardware feature giving means for direct and indirect addressing of stack, register, devices or other locations (local, external and externally indexed CPU memory).
Circle, Circular Arc, 3-D Cubics, Rectangle Generation	Firmware feature that generates instructions and data in refresh list to describe circle, arc, cubic curve, rectangle or square by interpreting parametric instructions provided in user-written display list.

Table 1-5. Performance Characteristics (Continued)

Item	Value
GPU Characteristics	
Arithmetic	Parallel 2's complement fractional
Data addresses	512 addressable GPU locations as follows: 247 RAM main R/W storage for user-addressable registers, option variables, and hardware stack features. 9 special addresses, addressed in RAM space 248 ROM addresses for constants and immediate values ("constants" ROM). 8 special registers, addressed in ROM space
Add time (GPU) Multiply time (GPU) Divide time (GPU)	250 nanoseconds (1 microcycle) 500 nanoseconds (2 microcycles) 1000 nanoseconds (4 microcycles)
Microinstruction ROM	24 bits per word, 4096 words maximum.
GPU internal interrupts	11 interrupts
Writable Control Store (WCS)	A hardware option which permits the user to modify, extend, or replace entirely, both the firmware implementing the user instruction set and the constants and immediate values normally stored in the "constants" ROM. This feature also provides the facility for the host computer to read the firmware code and constant values.
RBU Characteristics	
RBU size	8K 16-bit words standard; optionally expandable in increments of 8K words to a maximum of 32K words.
RBU cycle time	Sustained data transfers from RBU to DCU at a word rate of 900 nanoseconds.

Table 1-5. Performance Characteristics (Continued)

Item	Value
RBU Characteristics (Continued)	
RBU configurations	One RBU configuration accommodates all GPU optional features; the only RBU optional configuration is memory size, which may be used as single buffer, double buffer, or with the Edit "bubble."
DISPLAY CONTROLLER	The display control and display generation units consist of the Display Control Unit (DCU), the Vector Generator Unit (VGU), the Font Generator Unit (FGU), and the Monitor Control Unit (MCU).
Display interrupts	The DCU handles 4 device interrupts, a clock interrupt, a halt interrupt, and an end-of-list interrupt.
Refresh instruction set	6 basic refresh instructions (transparent to the user): CONTROL, LOAD, VECTOR ABS, VECTOR REL, VECTOR INCR, and CHARACTER.
Addressable registers	16 standard registers.
Frame rate	Programmable - 8 to 120 frames/second (synchronized with 60Hz input power).
Frame modes	Programmable - 7 frame modes as follows: CUTOFF, ALL CONTINUOUS, CLK (120 Hz if 60-Hz input power), SINGLE FRAME, OFF, and EXTERNAL SINGLE FRAME.
VECTORS	Absolute, relative, incremental, and incremental smoothed.
Addressable beam positions	4096X by 4096Y by 4096Z.
Drawing speed Linear draws Moves	20μs full scale (14" nominal). 0.75"/μs nominal + 1.33μs (reference Table 1-6). 1.33"/μs nominal + 1.33μs.

Table 1-5. Performance Characteristics (Continued)

Item	Value
VECTORS (Continued)	
Line texture	Programmable - 6 types as follows: Solid Lines, Long dashes, Short Dashes, Long/Short Dashes, Long/Short/Short Dashes, and Point Mode.
Smoothing	Programmable - generates curves through use of short 2D or 3D incremental vectors.
Intensity	256 programmable intensity levels in addition to Z range.
Blinking	Hardware blinking - displayed elements @ 2 Hz, cursor @ 4Hz.
Variable speed	A hardware feature which provides five programmable drawing rates varying from $1''/1.33\mu s$ maximum to $1''/500\mu s$ minimum.
Highlighting	A hardware feature which provides highlighting (brighten or blink and brighten) of selected or "Hit" elements.
CHARACTERS	
Aspect ratio	Height to width = 3:2.
Drawing technique	Analog - stroke method.
Character draw time	$5.5\mu s$ average per character including move to next character position.
Character size*	4 standard character sizes as follows: Size #1; 120 columns by 60 lines Size #2; 80 columns by 40 lines (default size) Size #3; 60 columns by 30 lines Size #4; 30 columns by 15 lines * Character size is a function of ROM constants and alternate sizing is available on request; e.g., 132 characters per line, 144 characters per line, etc.

Table 1-5. Performance Characteristics (Continued)

Item	Value
CHARACTERS (Continued)	
Font	2 types - normal and italics.
Subscript and superscript	Characters may be shifted up or down 1/4th line with an accompanying 1/3rd size reduction.
Programmable character set	Hardware option permits coding of up to 96 special characters.
90° rotation	Clockwise or counterclockwise rotation in 90° increments.
3D character rotation	Hardware option which provides continuous rotation about all three axes. Resolution = 1.5°/bit.
DISPLAY MONITOR	
CRT size	Rectangular CRT, 21" (53.34cm) standard.
Viewing area	16" wide by 14" high (40.64cm by 35.56cm). High accuracy area = 10" by 10" (25.4cm by 25.4cm).
Deflection	Electromagnetic
Focus	Electrostatic
Brightness	Greater than 6 foot-Lamberts measured on an isolated line at $1.33\mu s$/inch, 40Hz.
Contrast ratio	4:1 or greater.
Implosion protection	Bonded implosion shield.
Spot size	20 mil standard, 10 mil optional.

Table 1-5. Performance Characteristics (Continued)

Item	Value
DISPLAY MONITOR (Continued)	
Deflection resolution	4096 bits/14 inches = 293 points/inch. Display resolution = 50 lines per inch using shrinking raster method (100 lines per inch using 10 mil spot size option).
Repeatability	Less than 0.01 inch error for readdressing a point from any direction on the screen.
Linearity error	1% of full scale display along major axis.
Geometric distortion	Less than 2% error band for 10-inch horizontal or vertical vector.
Offscreen recovery time	Less than 0.01 inch error will occur when beam is returned to visible area at 1.33 μs/inch rate.
Jitter	0.005" maximum
Drift	0.05" in 8 hours after 30 minute warmup.
Operator controls	Front panel intensity and Focus controls are provided as operator controls. Servicing controls for X-Y centering and gain are accessible by removing the monitor cover.

1-5. VECTOR DRAWING RATES

Table 1-6 illustrates drawing times for various line lengths and the maximum number of lines which can be drawn at frame rates (FR) of 30 and 40Hz. Note that the maximum number of lines/frame is limited by both the RBU/DCU overhead per word(s) and the type of vector mode used (1, 2, or 3 words per vector draw). The following equations may be used to compute the drawing time and moving time for any length vector and the maximum number of lines per frame.

Draw time $= \text{Line length} \times 1.33\mu s + 1.33\mu s$

Move time $= \text{Length} \times 0.75 \times 10^{-6} + 1.33\mu s$

Lines/Frame $= 1/(\text{Frame Rate})(\text{Draw time})$

Table 1-6. Vector Drawing Rates

Vector Length	Draw Time Per Line*	Move Time	Vector Type	Maximum No. Lines/Frame	
				30Hz Refresh	40Hz Refresh
0.1"	1.47μs	1.41μs	2DI (1 word) 2D,3DI (2 words) 3D (3 words)	20,833 (1) 18,519 (2) 12,346 (3)	15,625 (1) 13,889 (2) . 9,259 (3)
0.2"	1.6μs	1.48μs	2DI 2D,3DI 3D	20,833 (1) 18,519 (2) 12,346 (3)	15,625 (1) 13,889 (2) 9,259 (3)
0.5"	2μs	1.71μs	2DI 2D,3DI 3D	16,667 16,667 12,346 (3)	12,500 12,500 9,259 (3)
0.75"	2.33μs	1.90μs	2DI 2D,3DI 3D	14,286 14,286 12,346 (3)	10,714 10,714 9,259 (3)
1.00"	2.67μs	2.08μs	2DI 2D,3DI 3D	12,500 12,500 12,346 (3)	9,375 9,375 9,259 (3)
1.5"	3.33μs	2.46μs	ALL	10,000	7,500
2.0"	4.00μs	2.83μs	ALL	8,333	6,250
4.0"	6.67μs	4.33μs	ALL	5,000	3,750
6.0"	9.33μs	5.83μs	ALL	3,571	2,679
8.0"	12.00μs	7.33μs	ALL	2,778	2,083
10.0"	14.47μs	8.83μs	ALL	2,273	1,705
12.0"	17.33μs	10.33μs	ALL	1,923	1,442
14.0"	20.00μs	11.83μs	ALL	1,667	1,250

* Includes time for DAC loading, gain switching, deceleration at end of draw, etc.

(1) Limited in 2-DI mode by 6.1μs RBU/DCU/VGU overhead.
(2) Limited in 2-D and 3-DI modes by 1.8μs RBU/DCU/VGU overhead.
(3) Limited in 3-D mode by 2.7μs RBU/DCU/VGU overhead.

SECTION II

FUNCTIONAL DESCRIPTION

2-1. INTRODUCTION

This section provides functional descriptions of the display system organization and each of the major functional units (refer to Figure 2-1). Definitions of special terms are contained in the glossary in Section VI.
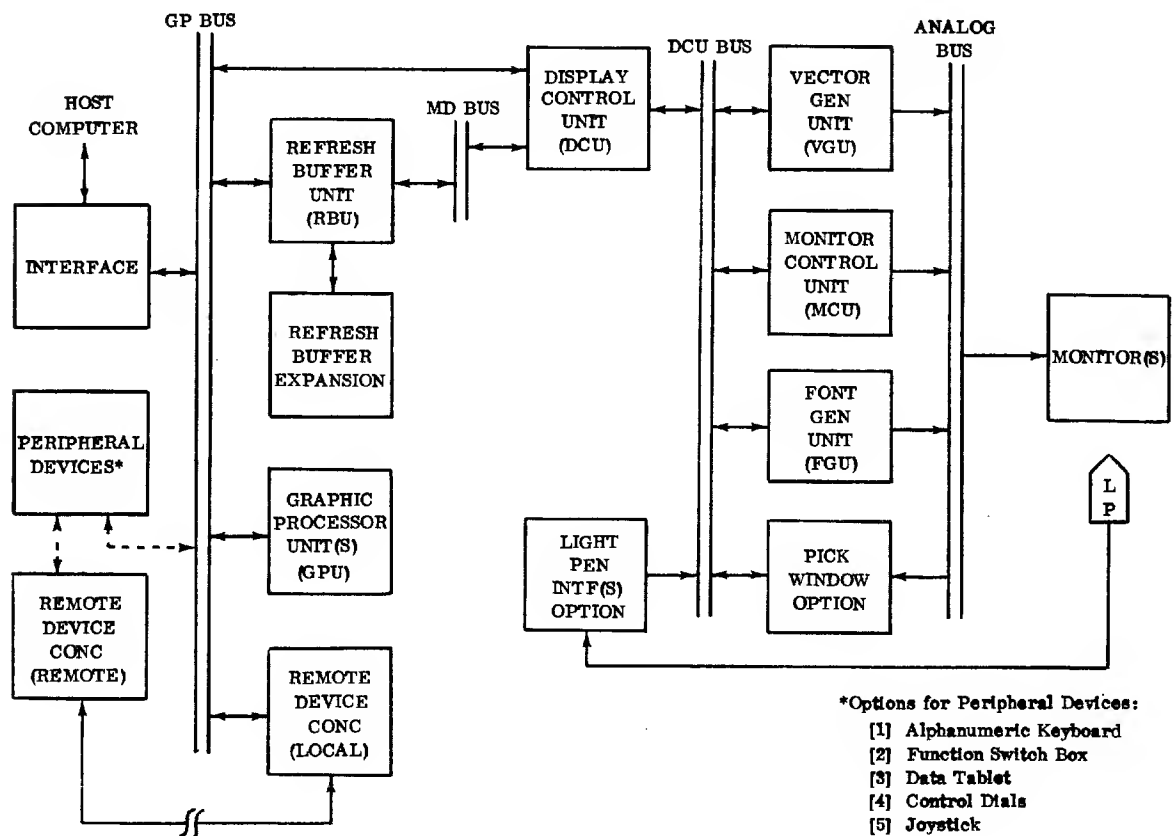
Figure 2-1. Display System Basic Block Diagram

2-2. DISPLAY SYSTEM ORGANIZATION

The block diagram illustrated in Figure 2-1 may be divided into four basic subsystems as follows:

>   GRAPHIC PROCESSOR UNIT (GPU)
>
>   DISPLAY CONTROLLER (RBU, DCU, VGU, FGU and MCU)
>
>   DISPLAY MONITOR
>
>   DISPLAY OPTIONAL DEVICES

In addition, Vector General offers a variety of interface assemblies to permit operation with virtually any host computer utilizing standard I/O techniques.

2-3. GRAPHIC PROCESSOR UNIT (GPU)

The GPU operates from a user instruction set comprised of 46 basic instructions (refer to Appendix A1 and Section III). In essence, it receives the display list consisting of instructions and associated data from the CPU, performs transformations and processing as required, and generates a display refresh list which is stored in the RBU. It is basically a high-speed special-purpose microprocessor designed to handle the most complex algorithms with ease. All transformations and image manipulations are performed digitally through use of a set of powerful microinstructions. In addition, interaction between the GPU and RBU provides the facility to permit element selection and picking and, through use of interactive devices, to tentatively alter the display list for purposes of CPU display list editing.

2-4. DISPLAY CONTROLLER

Generation and control of the displayed picture is performed by the RBU, DCU, VGU, FGU, and MCU (refer to Figure 2-1), which operate on the refresh list of instructions and data received from the GPU.

2-5. Refresh Buffer Unit (RBU)

The display refresh list stored in the RBU RAM may be continuously updated by the GPU. The list is accessed by the DCU in DMA fashion during each frame refresh cycle to update the displayed picture on the CRT screen. This may occur simultaneously with the generation of the following refresh list without display image interference. The RBU also contains control logic to move data from one part of memory to another when reorganizing the display refresh list for purposes of editing.

2-6. Display Control Unit (DCU)

The DCU uses the MD bus to access the refresh list from the RBU. It then processes the refresh instructions, routes the refresh data via the DCU bus to the VGU, FGU, and MCU, and generates the control signals required to display these data. It also contains logic to accommodate the Hit/Select option.

2-7. Vector Generator Unit (VGU)

The VGU, a high-speed vector generator which provides the deflection signals required to draw a line from one point to another on the face of the CRT, is located on one printed circuit board. The unit utilizes a closed-loop drawing system in which the time constant of the analog integrators is continuously changed in an infinite resolution arrangement to provide straight-line, high-accuracy vectors. Beam intensity is automatically adjusted as vector velocity changes during vector draws resulting in constant brightness on the CRT and thereby eliminating the need of any velocity programming requirement. The unit consists of D-to-A converters, gain control and switching circuits, multipliers, integrators, and the necessary control logic. The integrators retain the X-Y deflection voltages of the current beam position and generate a straight line draw (or move) between that point and the new endpoint specified by the following X-Y coordinates which are received from the DCU and applied to the D-to-A's. Data inputs from the DCU are applied to 2-level registers in the VGU thereby permitting simultaneous loading of new vector data while the VGU is operating on the previous data input. In addition to the deflection velocity signals required by the monitor during vector draws, the VGU also generates the unblanking and intensity level signals. The X-Y deflection voltages are passed directly to the monitor major deflection channels for X-Y deflection while the intensity and unblanking signals are passed, along with a deflection velocity signal, to the MCU where they are further processed for the monitor video channel. The VGU also utilizes a smoothing technique which, if specified by program during incremental vector modes, may be used to generate curved lines on the display. When the FGU is being used to display text, the VGU performs the spacing between character positions.

2-8.   Font Generator Unit (FGU)

The FGU contains all the circuitry required to generate the standard ASCII 96 char-
acter set for the display. It utilizes a programmed ROM in conjunction with "stroke"
character draws to provide crisp, high-speed characters regardless of the character
size selected by the program. The character codes, along with scaling, font, and
rotation parameters are received from the DCU via the DCU bus. In addition to the
standard 94 printable characters, a delete character and a blinking cursor may also
be drawn. Section V provides descriptions of the character instruction, character
word format, the control characters, and the usage of character scaling and rotation.
The FGU generates the unblanking signal and the deflection voltages for the monitor
minor deflection channels for all character draws; however, the VGU performs the
vector moves between the character positions.

2-9.   Monitor Control Unit (MCU)

The MCU is used to select the desired monitor for display and provides the unblank-
ing and intensity signals for the selected monitor video channel. Optional logic for
the Variable Speed Control and Color Monitor options is also installed on this circuit
board.

2-10.   DISPLAY MONITORS

The Model 3404 supports up to 4 CRT monitors (optionally up to 8). The standard
monitor is a high-speed refresh calligraphic CRT. An optional beam velocity con-
trol permits the selection of any of the following monitors: (a) low or high-speed
refresh CRTs, (b) storage tubes, and (c) scan converters. Various screen sizes
and phosphors are also available.

2-11.   DISPLAY DEVICES (OPTIONAL)

Numerous optional display devices are available for the Model 3404. These include
the alphanumeric keyboard, function switch box, light pen, data tablet, control dials,
and joystick. In addition, two features give added capability to the system. A pick
window circuit, normally working in conjunction with an optional device, provides
a movable window for generating interrupts. A remote device concentrator can be
employed when remote and local stations are physically situated between 25 feet and
600 feet apart. Its function is to concentrate data for serial transfer between local
and remote units.

Transfers of data to and from devices are normally performed under program control, either by the user program in the computer or by the firmware in the GPU. Most devices act solely as data sources. However, indicators associated with the function switch box can receive information that results in lamp illumination. Four devices generate actual computer interrupts that can be monitored by the program: keyboard, function switch box, light pen and data tablet.

SECTION III

GRAPHIC PROCESSOR PROGRAMMING

3-1.   INTRODUCTION

The Graphic Processor (GPU) is a microprocessor which, through use of a powerful
set of microinstructions, transforms instructions and data from the GPU display
list into a display refresh list comprising 2-D picture descriptions suitable for the
Display Monitor refresh. The display refresh list is first stored by the GPU in the
RBU and then accessed by the DCU during a display refresh cycle.

Operation of the RBU is asynchronous. This means that the GPU can generate a new
refresh list for the next refresh cycle and store it in another area of the RBU while
the DCU is refreshing the original RBU area. The DCU then accesses the new area
during the following refresh. The GPU and RBU also provide aids for interaction
and on-line display modification (see Section IV).

This section presents a delineation of display descriptions, user instruction formats,
addressing techniques, and GPU register organization. Refer to Section VI for des-
criptions of terms as they apply to the Model 3404 Graphics Display System.

Subjects described in this section are as follows:

3-2.   GPU OBJECTS

All display descriptions received by the GPU from CPU memory for processing are composed of Objects. Each object list is located through a Directory entry in computer memory and may occupy a logically contiguous area of memory (as opposed to physically adjacent memory locations). The GPU may process any number of these objects to generate the display refresh list to be used by the DCU for the final displayed picture. The first word of each object list contains a count of the number of words from the start of the object list to the first instruction to be processed (reference Figure 3-1).

Count of the number of words to the first instruction
LINKS
LOCAL OWN
INSTRUCTIONS
Return
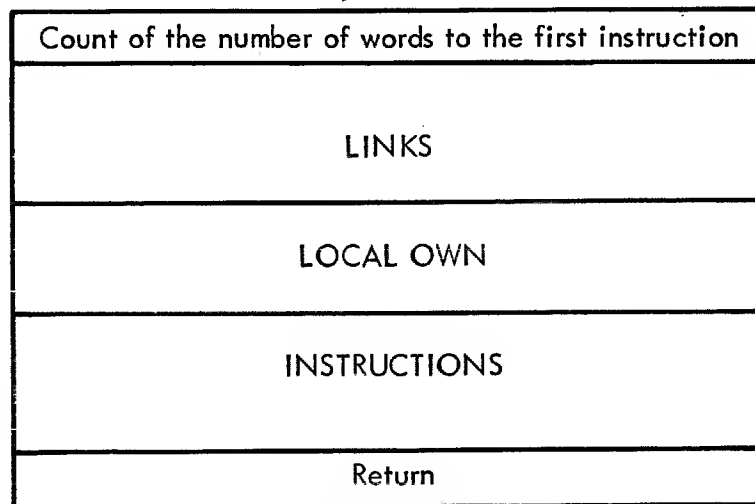
Figure 3-1. OBJECT List Structure

Each object list may contain the following three classes of information:

- LINKS   – The first part (second word and following) of each object may contain indices to Directory Entries for each independent object, or data areas referenced externally.

- LOCAL OWN   – The second part (after any LINKS) of each object may contain constants or own-variables [whose identity is the same for any (possibly recursive) invocation].

- INSTRUCTIONS – The third part of each object is the GPU instructions which describe the display to be generated by the object.

3-2

The last word of each object is a RETU instruction which either signals the end of output for the update pass or causes a return to a higher level calling object (reference RETURN instruction, paragraph 3-8).

3-3. GPU CONTROL

The GPU operation consists of "update" processing of display descriptions and optional Edit sequences. Aside from the instructions in the display list descriptions being processed, the GPU operation is governed by the following Parameter Registers and Commands:

GPU CONTROL PARAMETERS

- **Directory Address** — Location of the table in the host CPU used to locate all referenced objects and external variables.

- **Stack Boundaries** — The location and limit of the CPU memory available for nesting Object Calls, Environments (scale, rotation, etc.), Arguments, Temporary Variables, etc. This area in the host CPU is used only when the graphics RAM stack is filled.

- **Picture Base Object Number** — The directory index of the main object which directly defines or indirectly references all current desired views and their contents for display generation.

- **Edit Sequence** — The location and extent of words to insert, or to replace, a specific object portion for generation of an altered display output.

GPU COMMANDS

- **New Picture** — Initializes specified Picture, Object, Transform, Data Table, Element Generation, or Edit Functions.

- **Go** — Specifies execution or suspension of GPU processing.

- <u>Picture Halt Enable</u>   -Halts GPU processing at the conclusion of the specified picture.
NOTE: An interrupt to the CPU will also be generated if the Halt Interrupt Enable bit in the CONTROL (CTL) register is on. When zero, the GPU continuously updates from the user's display descriptions.

- <u>Same Frame</u>   -Allows one output frame to be built from independent input pictures under the current buffering or editing modes.

- <u>Devices Station#</u>   -This field is added to the high-order two bits of all addresses of user input devices (e.g., Data Tablet, Function Switches, Keyboard, Dials, Joystick) to select one group out of a maximum of four groups of devices.

- <u>DCU/RBU Station#</u>   -When multiple DCU/RBU pairs are used, this field specifies which pair is to receive the GPU output. It also designates that pair for processing its associated interaction-aid devices.

- <u>Buffer Mode</u>   -Specifies the RBU buffering mode (Single, Double or Editing modes).

- <u>Clip Text</u>   -Enables or disables clipping when processing character (TEXT instruction).

- <u>Clip Vector</u>   -Enables or disables clipping when processing Vectors (LINES instruction).

- <u>Initialize DCU</u>   -Causes the DCU to be initialized to the default conditions listed in Table 6-1.

3-4. <u>EFFECT OF USER INSTRUCTIONS ON GPU FUNCTIONS</u>

All functions implemented by the GPU are effected by the following six classes of operation and their argument parameters (reference Appendix A1 for table of user instructions):

- ELEMENT GENERATION   -These instructions are used to generate sets of lines, characters, or curves under current nested transformations. This class of instruction is described further in paragraph 3-5.

- LOAD VALUES

  -These instructions can move values between local, external, stack, or register areas. They can be used to load environmental changes (color, intensity, option control, etc.) and perform nonstandard or extended operations (i.e., compute list jumps or geometric constraints).

- CALL OBJECTS

  -These instructions perform "subroutine-like" invocations to other objects to generate an instance of the display defined by that object under current transformation and argument values.

- MAKE ARGS OR TEMPS

  -These instructions push "calling parameter" type arguments into the stack for use by invoked sub-objects, or store/reserve temporary variable storage in the stack for use by the current object.

- NEST ENVIRONMENT

  -These instructions are used to save current, and compose new, environmental parameters to affect the following element generation operations. These parameters are such values as scale, displacement, rotation, window arguments, etc.

- RETURN

  -These instructions "un-nest" the last saved environment and exit from a sub-object to return to the calling object, or to end the picture description if the instruction appears in the main object.

3-5. ELEMENT GENERATION

Instructions which specify visible display items may be classed into those using their arguments to generate lines, characters, or curves. The following list outlines some facilities implemented under each class:

LINES

- Vectors

  -Ordinary blanked or unblanked lines whose endpoints are specified by the instruction arguments.

LINES
(Continued)

- <u>Packed Tables</u>   -Vector sequences, and their control information, can be packed into byte or word machine boundaries. Also, data only (with no packed control) can be processed as arguments. This is the form in which all data are available via standard software, machine instructions, and I/O devices.

- <u>Graphs/Plots</u>   -Display instructions with one or two coordinates held fixed, or regularly incremented, can be packed as the only arbitrarily varying data. This improves memory utilization, definition processing, display filing, storing, and retrieving.

TEXT

- <u>Strings</u>   -Sequences of one or more alphanumeric characters or symbols at specified positions.

- <u>Blocks</u>   -A set of strings with a common left margin at a specified position.

- <u>Pages</u>   -Blocks of text positioned to fill the current transformed X/Y space in the Z = 0 plane.

CURVES OR LINE SEQUENCES

- <u>Rectangles</u>   -Arguments to this instruction specify the endpoints of a diagonal.

- <u>Circles</u>   -Arguments to this instruction specify a complete circle.

- <u>Arcs</u>   -Arguments to this instruction specify any portion of a circular arc.

- <u>Cubic</u>   -Arguments to this instruction specify endpoint and endpoint slopes of a third-order arc.

Table 3-1. User Instruction Set

Operation	Octal Code	Decimal Code	Hex Code	Mnemonic	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
CONTROL OPS	00000	0000	0000	NOOP	0	0	0	0												
	10000	4096	1000	RETU	0	0	0	1	0	0										
	14000	6144	1B00	RETZ	0	0	0	1	1	0										
	16000	7168	1C00	RETNZ	0	0	0	1	1	1										
	20000	8192	2000	GHALT	0	0	1	0												
	30000	12288	3000	BRKL	0	0	1	1	NUMBER OF WORDS											
DATA MOVE OPS	40000	16384	4000	LOAD	0	1	0	0	0	NUMBER OF VALUES										
	44000	18432	4800	LOADI	0	1	0	0	1	NUMBER OF VALUES										
	50000	20480	5000	NEST	0	1	0	1	0	REGISTER SET CODE										
	54000	22528	5800	NESTI	0	1	0	1	1	REGISTER SET CODE										
STACK OPS	60000	24576	6000	CALLU	0	1	0	0	0	OBJECT LINK INDEX										
	64000	26624	6800	CALLC	0	1	1	0	1	OBJECT LINK INDEX										
	70000	28672	7000	POP	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
	70000	28672	7000	PUSH	0	1	1	1	0	NUMBER OF VALUES										
	74000	30720	7800	GMARK	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
	74000	30720	7800	MPUSH	0	1	1	1	1	NUMBER OF VALUES										
ARITHMETIC OPS	100000	32768	8000	GADD	1	0	0	0	0								0	0	0	0
	100001	32769	8001	GSUB													0	0	0	1
	100002	32770	8002	GMPY													0	0	1	0
	100003	32771	8003	GDIV													0	0	1	1
	100004	32772	8004	GAND													0	1	0	0
	100005	32773	8005	GOR													0	1	0	1
	100006	32774	8006	GXOR													0	1	1	0
	100007	32775	8007	GSHFT					0								0	1	1	1
	104000	34816	8800	GADDI					1								0	0	0	0
	104001	34817	8801	GSUBI													0	0	0	1
	104002	34818	8802	GMPYI													0	0	1	0
	104003	34819	8803	GDIVI													0	0	1	1
	104004	34820	8804	GANDI													0	1	0	0
	104005	34821	8805	GORI													0	1	0	1
	104006	34822	8806	GXORI													0	1	1	0
	104007	34823	8807	GSHFTI	1	0	0	0	1								0	1	1	1
ELEMENT LIST OPS	130000	45056	B000	ARBI	1	0	1	1	0			NUMBER OF WORDS								
	134000	47104	B800	ARB	1	0	1	1	1	NUMBER OF WORDS										
	140000	49152	C000	LINES	1	1	0	0	LF		DF		BM		CLX		CLY		CLZ	
	160000	57344	E000	TEXT	1	1	1	0	LF		DF		PG		ROT		FNT		SZ	
SINGLE ELEMENT OPS	170000	61440	F000	CIRCLE	1	1	1	1	0								0	0	0	0
	170001	61441	F001	CCWARC													0	0	0	1
	170002	61442	F002	CWARC													0	0	1	0
	170003	61443	F003	RECT													0	0	1	1
	170004	61444	F004	CUBIC					0								0	1	0	0
	174000	63488	F800	CIRCL4					1								0	0	0	0
	174001	63489	F801	CCARC4													0	0	0	1
	174002	63490	F802	CWARC4													0	0	1	0
	174003	63491	F803	RECT4													0	0	1	1
	174004	63492	F804	CUBIC4	1	1	1	1	1								0	1	0	0

3-6.   USER INSTRUCTIONS

Table 3-1 provides a list of the GPU user instructions. The following paragraphs provide descriptions of the instruction fields, instruction usage, and the Refr addressing associated with each instruction (when applicable).

3-7.   NOOP INSTRUCTION (NOOP = 0000)

NOOP (0000)

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0	0	0	0												

The NOOP instruction causes no GPU operation and the GPU advances to the next instruction. NOOP does not affect the display list but it is counted as an element and as a picture word. The twelve LSBs may contain arbitrary data.

3-8.   RETURN INSTRUCTION (RETU=1000, RETZ=1800, RETNZ=1C00

The RETURN instructions terminate processing of the current object. If this is the main object of the picture-update processing, RETURN signals the end of output for this update pass. If the object was called by a higher level object, RETURN first clears the graphic stack of any temporary data (which may have been stored by a PUSH, GMARK or MPUSH instruction issued in the called object), then restores the calling object's environment to the GPU transformation, count and address registers. Processing then resumes in the higher level object after any saved transforms or temporary data (not protected by a GMARK) are restored or cleared from the calling object's stack. (Also reference NEST and STACK instructions.)

RETU (1000)

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0	0	0	1	0	0										

The RETU instruction executes an unconditional return.

RETZ (1800)

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0	0	0	1	1	0										

The RETZ instruction executes a return if the contents of register GP1 are zero; otherwise a NOOP is performed.

RETNZ (1C00)

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0	0	0	1	1	1										

The RETNZ instruction executes a return if the contents of register GP1 are non-zero; otherwise a NOOP is performed.

3-9.   GHALT INSTRUCTION (2000)

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0	0	1	0												

This instruction halts GPU processing and places the GPU in the "Halt-Instruction"
state. If the GPU halt-interrupt enable bit is set (CTL[15]), the GPU generates an
interrupt to the CPU. GPU processing will resume at the next instruction when a
GO command (CMD[01]) without NPIC (CMD[00]) is issued. If, instead, a GO command
and NPIC are both issued, processing will begin at the first instruction of the Picture
Base Object.

3-10.   BRKL INSTRUCTIONS (BRKLS = 3000, BRKLX = 3FFF)

The break-list instruction allows a set of discontiguous buffer areas to be linked together by specifying either of the two forms of the BRKL instruction (BRKLS or BRKLX) as shown in the following diagrams.

BRKLS (3000)

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0	0	1	1	RELATIVE # OF WORDS											

The contents of bits 04 through 15, if not all ones, specify a 12-bit 2's complement "word displacement" (bit 04 is taken as the sign bit) to be added to the word address of the word following the BRKLS instruction in order to locate the next GPU instruction to be executed. This short form of the instruction limits at a range of -2048 to +2047 from the address of the next word.

BRKLX (3FFF)

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RELATIVE # OF WORDS															

If bits 04 through 15 are all ones, the contents of the word following the BRKLX instruction specify a 16-bit 2's complement "word displacement" to be added to the word address of the word following the BRKLX instruction in order to locate the next GPU instruction to be executed.

NOTE: BRKL is not counted as a word in PWC nor as an element in ELN. As a result, object processing is transparent to the existence of any number of such breaks.

3-11. LOAD INSTRUCTIONS (LOAD = 4000, LOADI = 4800)

These instructions, depending on the state of bit 04, may have one of two formats LOAD or LOADI) as shown in the following diagrams.

LOAD (4000)

00	01	02	03	04	05 06 07 08 09 10 11 12 13 14 15
0	1	0	0	0	n
Destination Refr					
List of n Source Refrs					
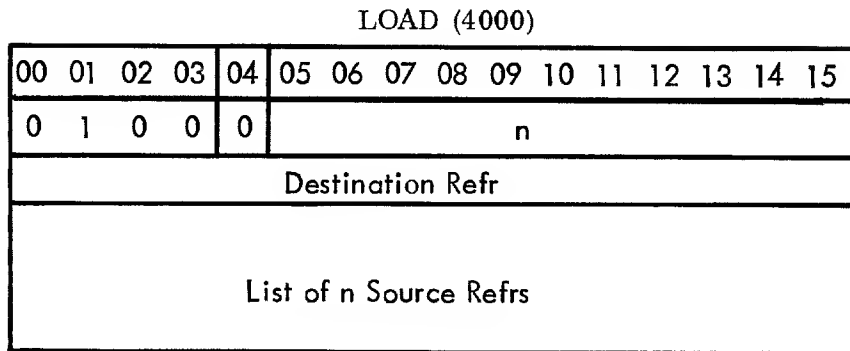
The LOAD instruction transfers "n" referenced values to sequential locations, beginning with the specified destination (base address). Note that the list of "n" Source Refrs contains the address of each operand, not the operand itself.

LOADI (4800)

00	01	02	03	04	05 06 07 08 09 10 11 12 13 14 15
0	1	0	0	1	n
Destination Refr					
List of "n" values					
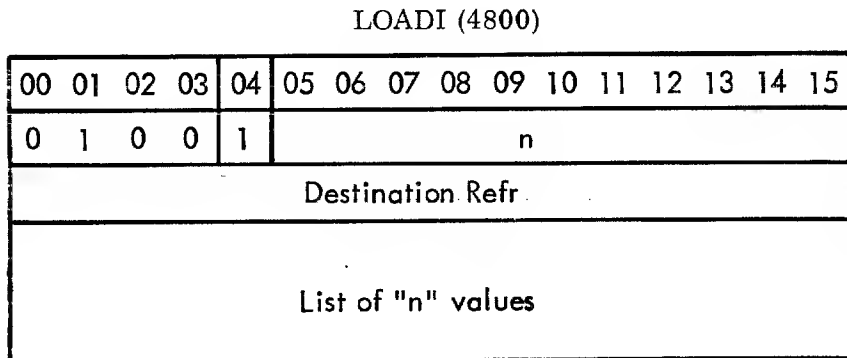
The LOAD IMMEDIATE instruction (LOADI) similarly transfers "n" values from the data list to sequential locations starting at the Destination Refr. Note that the list shown above contains the actual operands.

3-12. NEST INSTRUCTIONS (NEST = 5000, NESTI = 5800)

The nest instructions, depending on the state of bit 04, may have one of two formats (NEST or NESTI) as shown in the following diagrams. Note that in both cases the actual registers affected (and their number) are specified by the Register Set Code (reference Table 3-2).

NEST (5000)

00	01	02	03	04	05 06 07 08 09 10 11 12 13 14 15
0	1	0	1	0	Register Set Code
Refrs					

One method for efficiently altering environmental parameters of objects is through the use of the NEST instructions. The required "Refr" values following the NEST instruction word (see #ARGS in Table 3-2) are used to change a corresponding number of environmental registers. However, prior to such action, the current contents of these designated registers are stored in a stack to enable restoration of the current environment after a subsequent CALL when either a RETURN or POP instruction is executed. In summary, a NEST instruction results in these events:

a) storage of current contents of transformation registers defined by Register Set Code into stack. A control word is also saved in the stack for control of later retrieval.

b) composition of new values for transformation registers by applying arguments specified by Refrs to current transform values.

c) Loading of new values into registers specified by Register Set Code.

NESTI (5800)

00	01	02	03	04	05 06 07 08 09 10 11 12 13 14 15
0	1	0	1	1	Register Set Code
Immediate Values				

The NEST IMMEDIATE instruction (NESTI) functions in a similar manner as the NEST instruction. However, immediate values in the NESTI instructions are actual operands, while the Refr values in NEST instructions represent addresses of operands.

NOTE: Nested function NORXYZ (code 04) is interpreted as follows. The rotation matrix applied by an RXYZ nest operation is equivalent to an X rotation matrix applied to a Y rotation matrix and further applied to a Z rotation matrix.
    The effect of applying this resultant matrix to a vector is equivalent to rotating the vector about its Z-axis by a z-rotation angle. Then, the result is rotated about its new Y-axis by a y-rotation angle. Finally, this result is rotated about its X axis by the x-rotation angle.

Table 3-2. Register Set Codes For Nesting Values

Code			Mnemonic	#ARGS	N/M	Description
DEC	HEX	OCT				
00	00	00	NOSXY	3	N	2D object scale, X, Y displacements (OS,ODX,ODY)
01	01	01	NOSXYZ	4	N	3D object scale, X, Y, Z displacements (OS,ODX,ODY,ODZ)
02	02	02	NODXY	2	N	2D object X, Y displacements (ODX,ODY)
03	03	03	NODXYZ	3	N	3D object X, Y, Z displacements (ODX,ODY,ODZ)
04	04	04	NORXYZ	3	N	3D rotation, Z then Y then X (RZ,RY,RX)
05	05	05	NORZYZ	3	N	3D rotation, Z then Y then Z (Euler angles) (RZ,RY,RZ)
06	06	06	NOS	1	N	Object scale (OS)
07	07	07	NODX	1	N	Object X displacement (ODX)
08	08	10	NODY	1	N	Object Y displacement (ODY)
09	09	11	NODZ	1	N	Object Z displacement (ODZ)
10	0A	12	NRX	1	N	X rotation value (RX)
11	0B	13	NRY	1	N	Y rotation value (RY)
12	0C	14	NRZ	1	N	Z rotation value (RZ)
13	0D	15	MPSIXY	4	M	Picture scale, intensity scale, X, Y displacements (PS,PDI,PDX,PDY)
14	0E	16	MWCXYS	5	M	Window center point, near Z, size X, size Y (WCX,WCY,WNZ,WSX,WSY)
15	0F	17	MPDXY	2	M	Picture X, Y displacements (PDX,PDY)
16	10	20	MWCXY	2	M	Window center point (WCX,WCY)
17	11	21	MPS	1	M	Picture scale (PS)
18	12	22	MPDX	1	M	Picture X displacement (PDX)
19	13	23	MPDY	1	M	Picture Y displacement (PDY)
20	14	24	MWCX	1	M	Window X center point (WCX)
21	15	25	MWCY	1	M	Window Y center point (WCY)
22	16	26	MWS	2	M	Window size (WSX,WSY)

NOTES: The above codes are used for NEST and NESTI instructions only.

#ARGS = The number of arguments used in the modifications.

M = The current values of the specified registers are first stacked and are then replaced with new argument values.

N = The current values of the specified registers are first stacked after which new values are composed (by applying the arguments to the current values) and loaded into the specified registers.

3-13.   CALL INSTRUCTIONS (CALLU = 6000, CALLC = 6800)

CALLU (6000)

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0	1	1	0	0	Object Link Index										

CALLC (6800)

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0	1	1	0	1	Object Link Index										

The CALL instruction, depending on the state of bit 04, may have one of two formats, an unconditional call (CALLU) or a conditional call (CALLC). If bit 04 is a zero, an unconditional jump is made to the called object. If bit 04 is a one and the contents of GPU general purpose register GP1 = 0, a NOOP occurs (after discarding or restoring any stack contents as a RETU would have done).

The CALL instruction establishes the object number contained in the LINK area of the current object, indexed by the Object Link Index, as the next lower level object to be processed after saving the current environment. This accomplishes a sub-picture call to output, as a single element of the current object, all of the picture described by the object being called and any sub-objects it calls. Saving of the environment necessary to return and process the remainder of the current object is accomplished by stacking the current SA, ELN, OBA, OBN and IA register values. The stack base SA is then set to the final stack top ST, thus establishing a fresh empty stack area for the called object.

3-14.   STACK INSTRUCTIONS (POP, PUSH=7000, GMARK, MPUSH=7800)

Four "Stack" instructions (POP, PUSH, GMARK, and MPUSH) are provided to:

a)   Move argument or initial temporary values to the stack for program availability and to remove and discard these same values when the current object is exited.

b)   Protect local stacks across CALLs.

c)   Clear the local stack.

d)   Restore nested environment.

The instruction formats and descriptions of their usage are shown in the following drawings.

POP (7000)

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

The stack instruction (POP) removes all temporaries (stored by PUSH or MPUSH) and nested values (stored by NEST) upto and including the first MARK (or MPUSH) encountered in the graphic stack.

PUSH (7000)

00	01	02	03	04	05 06 07 08 09 10 11 12 13 14 15
0	1	1	1	0	Number of values
					Refrs

The MOVE ARGUMENTS OR TEMPORARIES instruction (PUSH), followed by reference addresses, moves temporary values or argument values to be passed by the current object to the stack for later program availability. The values are obtained from the addresses of the list of Refrs following the instruction. The stored list of values is followed by a stack control word which allows the list of stacked temporaries to be discarded when the current object is exited by a RETURN instruction issued in this object or in the object that it calls, or by a POP instruction issued in this object.

GMARK (7800)

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

The GMARK instruction puts a one-word stack marker in the stack to protect "pushed" or "nested" items across subsequent calls. The stack marker and any previously "pushed" or nested" items remain in the stack until removed by a POP or RETURN instruction issued in the current object.

MPUSH (7800)

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0	1	1	1	1	Number of values										
Refrs															

The "mark and push" instruction (MPUSH) is identical with the PUSH instruction previously described with the exception that a stack marker is first put in stack before the user-specified temporaries and the stack control word. This permits previously stacked items to remain in the stack even if the current object issues a call to another object, which may then perform a return (see GMARK instruction).

3-15.   ARITHMETIC INSTRUCTIONS (ARITH = 8000, ARITHI = 8800)

Eight arithmetic operations, performed either on two referenced values or on one referenced and one immediate value, may be specified by the following arithmetic instruction.

00 01 02 03	04	05 06 07 08 09 10 11	12 13 14 15
1 0 0 0	1		OP

The OP field (bits 12 through 15) specifies the type of operation which is to be performed on the first two words following the instruction, the result of which is stored at the reference address specified by the third word. The I field (immediate field bit 04) when = 0 specifies that the second word following the instruction is a referenced value, and when = 1 specifies the same word as an immediate value. The two forms of the instruction are shown in the following diagram.

ARITH (8000)

00 01 02 03	04	05 06 07 08 09 10 11	12 13 14 15
1 0 0 0	0		OP
Refr A			
Refr B			
Refr C			

The ARITH instruction specifies that the operation (OP, reference Table 3-3) is to be performed on the two values at reference addresses A and B and that the result is to be placed in reference address C.

ARITHI (8800)

00 01 02 03	04	05 06 07 08 09 10 11	12 13 14 15
1 0 0 0	1		OP
Refr A			
Value B			
Refr C			

The ARITHI instruction specifies that the operation (OP) is to be performed on the value at reference address A and the immediate value B with the result placed at reference address C.

The OP field is decoded to provide arithmetic operations as illustrated in Table 3-3 (mnemonics and descriptions for both "reference" and "immediate" are provided).

Table 3-3. Arithmetic Operations

OP Field	Mnemonic	Referenced OP	Mnemonic	Immediate OP
0	GADD	$C \leftarrow A + B$	GADDI	$C \leftarrow A + \text{Value } B$
1	GSUB	$C \leftarrow A - B$	GSUBI	$C \leftarrow A - \text{Value } B$
2	GMPY	$C \leftarrow A \times B$	GMPYI	$C \leftarrow A \times \text{Value } B$
3	GDIV	$C \leftarrow A \div B$	GDIVI	$C \leftarrow A \div \text{Value } B$
4	GAND	$C \leftarrow A \wedge B$	GANDI	$C \leftarrow A \wedge \text{Value } B$
5	GOR	$C \leftarrow A \vee B$	GORI	$C \leftarrow A \vee \text{Value } B$
6	GXOR	$C \leftarrow A \veebar B$	GXORI	$C \leftarrow A \veebar \text{Value } B$
7	GSHFT	$C \leftarrow A \times 2^{B}$	GSHFTI	$C \leftarrow A \times 2^{\text{Value } B}$

NOTES: 1. Values are taken as 2's complemented, 16-bit fractions (binary point is at the right of the leading sign bit). Examples are shown below.

     C000 – One-half full-scale negative

     8000 – Full-scale negative

     7FFF – Full-scale positive

     4000 – 1/2 Full-scale positive

    2. Overflow conditions are ignored.

3-16.   ARB INSTRUCTIONS (ARBI = B000, ARB = B800)

ARBITRARY instructions (ARB and ARBI) function to insert arbitrary sequences of unmodified words into the RBU via the GPU. These unmodified words can consist of several instructions and their associated data or might possibly consist of a single instruction without any accompanying data. Examples of instructions that can be transferred to the RBU are shown in Figure 5-1. The CONTROL instruction consists of a single word, while all LOAD, VECTOR and CHARACTER instructions require both an instruction word plus data.

Bit 04 of the ARBITRARY instruction determines whether the instruction is an ARBITRARY or ARBITRARY IMMEDIATE. The ARB instruction contains a single word "Refr" which represents the address of the first word to be transferred to the RBU (first word of a table). The total number of words to be transferred is contained in the least-significant portion of the ARBITRARY instruction (bits 08 through 15, inclusive). The format for an ARB instruction is shown below.

ARB (B800)

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	0	1	1	1				NUMBER OF WORDS							
Refr															

The ARBITRARY IMMEDIATE instruction (ARBI) contains all instructions and assoc-
iated data in the words immediately following the ARBI command. As described above,
"n" can be 1 (as in the case of a single CONTROL instruction) or can be another posi-
tive integer. For example, a CHARACTER instruction plus 1-word data (2 words) and LOAD
instruction plus 6-word data (7 words total) would occupy a space of 9 words in the
ARBI instruction. Thus, NUMBER OF WORDS field (bits 8-15) equals a +9. Refer
to Paragraphs 5-9 through 5-17 for information concerning these RBU instructions.
The format for an ARBI instruction is shown below.

ARBI (B000)

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	0	1	1	0				NUMBER OF WORDS							
n words															

3-17.   LINES INSTRUCTION (C000)

00 01 02 03	04 05	06 07	08 09	10 11	12 13	14 15
1 1 0 0	LF	DF	BM	CLX	CLY	CLZ
Values or References						

The LINES instruction is used to generate a list of line elements. Descriptions of the fields and their effects on the words following the instruction are provided below.

3-18.   LF Field (Bits 4 and 5)

The List Format field (LF) has four possible values:

00   LFIT   Immediate list of values with terminate in last value of last element.
01   LFRT   Reference to list of values with terminate in last value of last element.
10   LFIC   Reference to element count, immediate list of values.
11   LFRC   Reference to element count , reference to list of values.

LFIT   (Immediate With Terminate) All element generation parameters are in
            a data list following the instruction as shown below. The list termination
            is encoded within the data fields (possibly packed).

LFIT   Immediate With Terminate   (0000)

00 01 02 03	04 05	06 07 08 09 10 11 12 13 14 15
1 1 0 0	0 0	
List of values with LSB = 0		
Last value of last element with LSB = 1		

LFRT   (Referenced With Terminate) This instruction is followed by a reference which gives the location of the instruction's data list. The list contains data values (possibly packed) with "list termination" encoded with the values.

LFRT   Referenced With Terminate   (0400)

00	01	02	03	04	05	06 07 08 09 10 11 12 13 14 15
1	1	0	0	0	1	
Refr to list of values with terminate					

LFIC   (Count, Immediate List) This instruction is followed by the count (or refer to it) which is immediately followed by the list itself. The list of values (possibly packed) does not require that any control information be kept with these values.

LFIC   Count, Immediate List   (0800)

00	01	02	03	04	05	06 07 08 09 10 11 12 13 14 15
1	1	0	0	1	0	
Refr to length "n"					
List of "n" values					

LFRC   (Count, Referenced List) This instruction is followed by two references which give the length of the list and its location. The list contains data only (possibly packed).

LFRC   Count, Referenced List   (0C00)

00	01	02	03	04	05	06 07 08 09 10 11 12 13 14 15
1	1	0	0	1	1	
Refr to length "n"					
Refr to list of "n" values					

3-19.   DF Field (Bits 6 and 7)

The Data Format field (DF), comprised of bits 6 and 7 of the LINES instruction, has four possible states:

00	DFWD	Full word
01	DFBY	Byte
10	DFB4	Byte/4
11	DFRF	Reference

The DF field is utilized to specify the format of coordinate values. These values will follow the instruction word or be contained in an addressed Table (see LF field on pages 3-23 and 3-24 for possibilities). In the case of the DFWD, DFBY and DFB4 states, the associated values are given in left-justified 2's complement fractional notation. If the Reference With Terminate feature is utilized (see LFRT on page 3-24), bit 15 (LSB) of any 16-bit word in the list of values indicates a list terminate condition when it equals a 1.

In the case of the DFRF option for the DF field, the list contains reference addresses or reference data in the format shown in Table 3-5. If immediate positive or immediate negative values are used, the words represent data with 13-bits of magnitude (the three most significant bits must be either all zeros or all ones). If any of the other nine combinations of Table 3-5 are used, the words represent addresses to reference the data.

DFWD   (Full Word) Each successive value is taken from a full 16-bit word in the list. Since two's complement fractional notation is used, typical values are as follows:

C000   - One-half full-scale negative
8000   - Full-scale negative
7FFF   - Full-scale positive
4000   - 1/2 Full-scale positive

DFWD   Full Word   (0000)

00 01 02 03	04 05	06 07	08 09 10 11 12 13 14 15
1 1 0 0		0 0	

Value
Value
⋮
Value "n"

DFBY   (Byte) Each value is taken from successive bytes in the list, packed as two bytes per word. The coordinate information is taken by left-justifying each data byte so that its leading bit is in the sign position. If termination is encoded in the list, the lowest bit of each byte is used to signal list terminate when = 1.

The order of bytes in a word corresponds to the assembler convention of the host computer, defined by the GPU BYTE 12 condition (BYTESWAP signal on GP Bus).

DFBY   Byte   (0100)

00 01 02 03	04 05	06 07	08 09 10 11 12 13 14 15
1 1 0 0		0 1	

Value	Value
Value	Value
Value	Value

This format is convenient for packing constrained or natural data when it only has 8 or less significant bits of resolution. This reduces the amount of data processed or stored by the user and improves the display processor speed. When the 1/256th of full scale maximum resolution proves too coarse for the desired display, the following DFBA format permits resolution down to 1/1026th of full scale. This is gained at the expense of the maximum attainable range per value (1/4th of full scale).

DFB4   (Byte/4) Each value, shown in Byte form, is divided by 4 (2 right shifts with sign extension) prior to usage as coordinates of LINES instruction. If Reference With Terminate is used (see LFRT on page 3-24), the setting of the lowest bit of any byte will terminate the list.

DFB4		Byte/4			(0200)

00 01 02 03	04 05	06 07	08 09 10 11 12 13 14 15
1 1 0 0		1 0	

4 x Value	4 x Value
4 x Value	4 x Value
⋮	⋮
4 x Value	4 x Value

DFRF   (Reference) Each word on the list is used as a "Refr" to specify either an immediate value or the reference address of a value (see Table 3-5 for format of Refr).

DFRF		Reference			(0300)

00 01 02 03	04 05	06 07	08 09 10 11 12 13 14 15
1 1 0 0		1 1	

Refr
Refr
Refr
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Value References

NOTE: Due to look-ahead buffering of data in the DFRF (DF=11) format, the list must be followed by a NOOP instruction (word of zeros).

3-27

3-20.   <u>BM Field</u> (Bits 8 and 9)

The Beam Sequence field (BM), comprised of bits 8 and 9 of the LINES instruction, specifies the sequence of controls for the processing of Line Type lists as follows:

00 BMDJ   Disjoint line string
01 BMJL   Joined line string
10 BMHV   Horizontal/vertical run
11 BMPT   Points
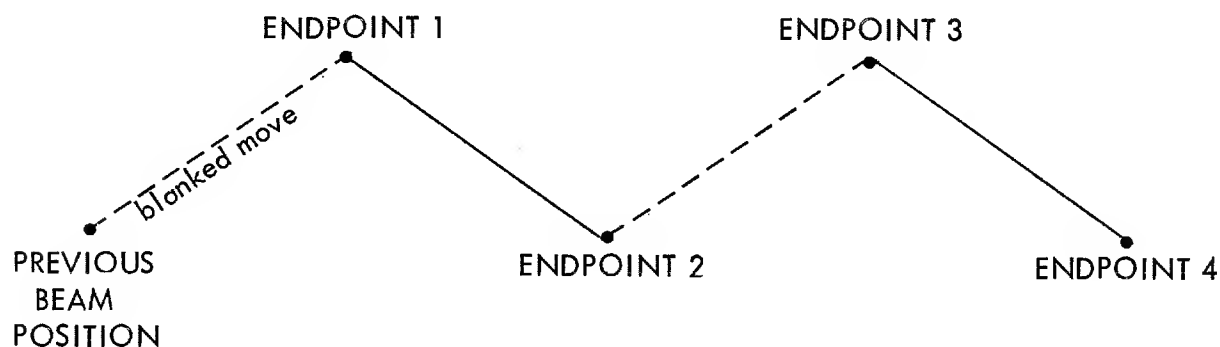
Figure 3-2. Disjoint Line Strings

BMDJ   (Disjoint Line Strings) The beam alternates between "move" and "draw" to successive coordinate set endpoints beginning with a "move" to the first endpoint (see Figure 3-2).

BMDJ   (0000)

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	1	0	0					0	0						
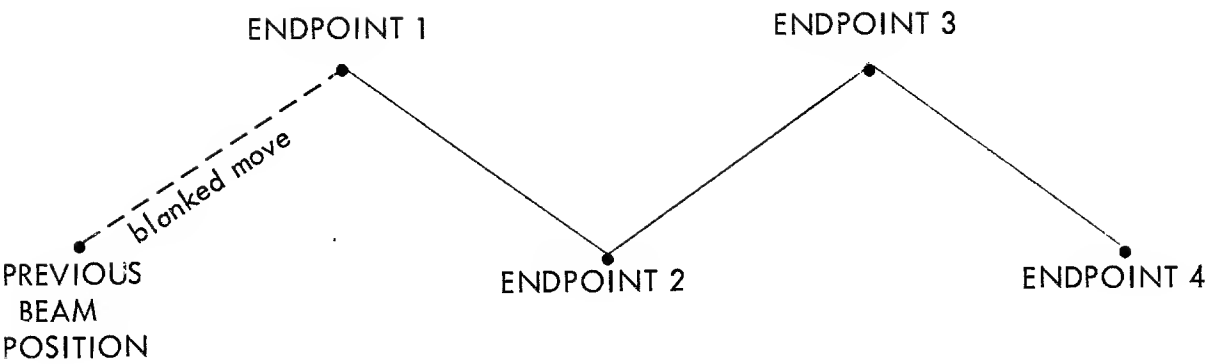
Figure 3-3. Joined Line String

BMJL   (Joined Line String) The beam does an initial "move" to the first coordinate set followed by "draws" to remaining sets in the list (see Figure 3-3).

BMJL   (0040)

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	1	0	0					0	1						
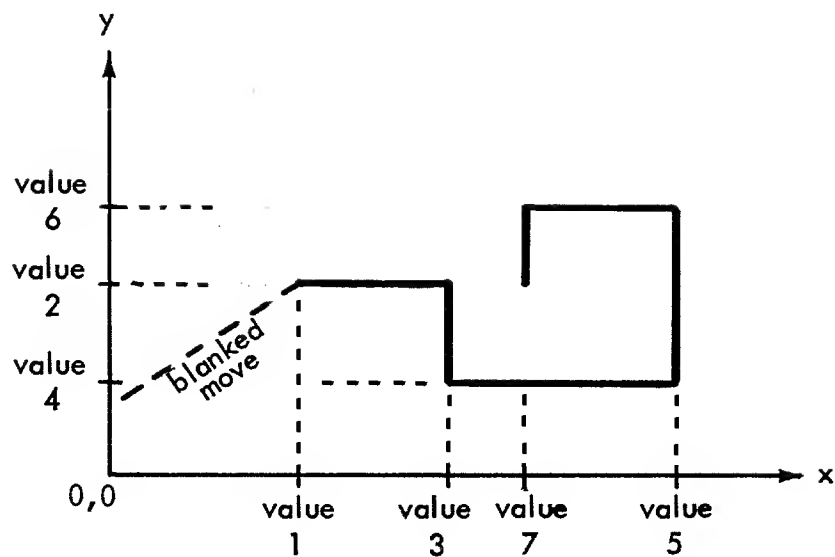
Figure 3-4. Horizontal/Vertical Run

BMHV   (Horizontal/Vertical Runs) The beam control is the same as above except that CLX and CLY field effects are alternated after each "draw" (see Figure 3-4.)

BMHV   Horizontal/Vertical Run   (0080)

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	1	0	0					1	0						

Data List

Value 1
Value 2
Value 3
Value 4
Value 5
Value 6
Value 7
Value 2

ENDPOINT 2
•

ENDPOINT 4
•

ENDPOINT 1
•

ENDPOINT 3
•
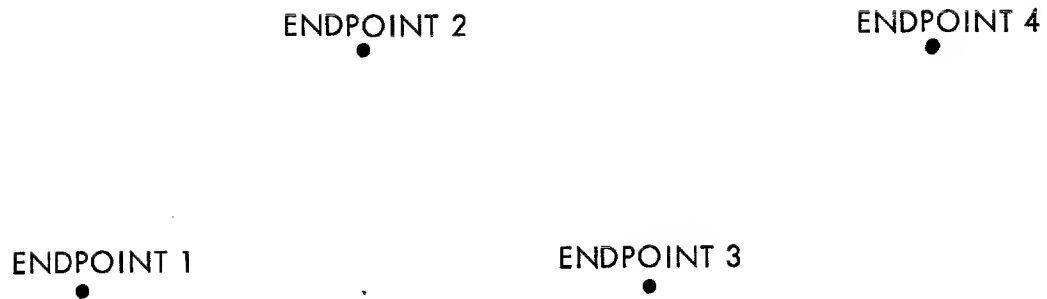
Figure 3-5.   Points Plotted Using BMPT Mode

BMPT   (Points) Each coordinate set (including the first set) defines the location
            of a point to be plotted.   This specification overrides the "Line Type"
            field of the LNCT register but does not change LNCT (see Figure 3-5).

BMPT   Points   (00C0)

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	1	0	0					1	1						

3-21.   CLX Field (Bits 10-11)

The X-Coordinate Loading field (CLX), comprised of bits 10 and 11 of the LINES instruction, specifies how successive coordinate values from the list are to be used to affect coordinate register X. A description of the format is shown below.

CLX Field

00	CCX	(Constant) This specifies that the coordinate is constant, and the updating of this coordinate is to be skipped (the list supplies no data for it). At most, only two fields can be zero.
01	CIX	(Stepped) This specifies that the coordinate field is to be incremented by increment register DLTX after each beam operation. The list supplies no data for the coordinate.
10	CAX	(Absolute) This causes the next list value to replace the contents of the X coordinate register.
11	CRX	(Relative) This causes the next list value to be added to the current contents of the X coordinate register. Any overflow condition created by the addition is ignored. (The value will "wrap around" the edge of the local object number space.)

CCX   Constant X Coordinate   (0000)

X ⟵ X

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	1	0	0							0	0				

CIX   Stepped X Coordinate   (0010)

X ⟵ X + DLTX

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	1	0	0							0	1				

CAX   Absolute X Coordinate   (0020)

X ⟵ Value

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	1	0	0							1	0				

CRX   Relative X Coordinate   (0030)

X ⟵ X + Value

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	1	0	0							1	1				

3-22.　CLY Field (Bits 12-13)

The Y-Coordinate Loading field (CLY), comprised of bits 12 and 13 of the LINES instruction, specifies how successive coordinate values from the list are to be used to affect coordinate register Y.　Formats are shown below.

CLY Field

00	CCY	(Constant) This specifies that the coordinate is constant, and the updating of this coordinate is to be skipped (the list supplies no data for it. At most, only two fields can be zero.
01	CIY	(Stepped) This specifies that the coordinate field is to be incremented by increment register DLTY after each beam operation. The list supplies no data for the coordinate.
10	CAY	(Absolute) This causes the next list value to replace the contents of the Y coordinate register.
11	CRY	(Relative) This causes the next list value to be added to the current contents of the Y coordinate register. Any overflow condition created by the addition is ignored. (The value will "wrap around" the edge of the local object number space).

CCY　　　　　　Constant Y Coordinate　　　　　(0000)

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	1	0	0									0	0		

Y ⟵ Y

CIY　　　　　　Stepped Y Coordinate　　　　　(0004)

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	1	0	0									0	1		

Y ⟵ Y + DLTY

CAY　　　　　　Absolute Y Coordinate　　　　(0008)

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	1	0	0									1	0		

Y ⟵ Value

CRY　　　　　　Relative Y Coordinate　　　　(000C)

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	1	0	0									1	1		

Y ⟵ Y + Value

3-23.   **CLZ Field** (Bits 10-11)

The Z-Coordinate Loading field (CLZ), comprised of bits 14 and 15 of the LINES
instruction, specifies how successive coordinate values from the list are to be used
to affect coordinate register Z. A description of the format is shown below.

CLZ Field

00	CCZ	(Constant) This specifies that the coordinate is constant, and the updating of this coordinate is to be skipped (the list supplies no data for it. At most, only two fields can be zero.
01	CIZ	(Stepped) This specifies that the coordinate field is to be incremented by increment register DLTZ after each beam operation. The list supplies no data for the coordinate.
10	CAZ	(Absolute) This causes the next list value to replace the contents of the Z coordinate register.
11	CRZ	(Relative) This causes the next list value to be added to the current contents of the Z coordinate register. Any overflow condition created by the addition is ignored. (The value will "wrap around" the edge of the local object number space.)

CCZ — Constant Z Coordinate (0000)

Z ← Z

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	1	0	0											0	0

CIZ — Stepped Z Coordinate (0001)

Z ← Z + DLTZ

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	1	0	0											0	1

CAZ — Absolute Z Coordinate (0002)

Z ← Value

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	1	0	0											1	0

CRZ — Relative Z Coordinate (0003)

Z ← Z + Value

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	1	0	0											1	1

3-24. TEXT INSTRUCTIONS (E000)

00 01 02 03	04 05	06 07	08 09	10 11	12 13	14 15
1 1 1 0	LF	DF	PG	ROT	FNT	SZ
Values or References						

This instruction is used to generate lists of text elements. Descriptions of the fields and their effects on the words following the instructions are provided below (refer also to Appendix A5 for the standard character set and user control codes).

3-25. <u>LF Field</u> (Bits 4 and 5)

The List Format field (LF) comprised of bits 4 and 5 of the TEXT instruction is identical to that described in the LINES instruction. The LF field has four possible modes:

00	LFIT	Immediate list of values with terminate in last value of last element.
01	LFRT	Reference to list of values with terminate in last value of last element.
10	LFIC	Reference to element count, immediate list of values.
11	LFRC	Reference to element count, reference to list of values.

LFIT   (Immediate With Terminate) All element generation parameters are in a data list following the instruction. The list termination is encoded within the data fields (possible packed).

LFIT   Immediate With Terminate   (0000)

00 01 02 03	04 05	06 07 08 09 10 11 12 13 14 15
1 1 1 0	0 0	
List of Character Data, Terminated by TRM Character (=9C)		

LFRT   (Referenced With Terminate) This instruction is followed by a reference which gives the location of the instruction's list. The list contains data values (possibly packed) with "list termination" enclosed with the values.

LFRT   Referenced With Terminate   (0400)

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	1	1	0	0	1										
Reference to "A"															

Location "A"

List of Character Data,
Terminated by TRM Character (=9C)

LFIC   (Count, Immediate List) This instruction is followed by the count (or reference to it) which is immediately followed by the list itself. The list of values (possibly packed) does not require that any control information be kept with these values.

LFIC   Count, Immediate List   (0800)

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	1	1	0	1	0										
n															
List of "n" characters															

LFRC   (Count, Referenced List) This instruction is followed by two references which give the length of the list and its location. The list contains data only (possibly packed).

LFRC						Count, Referenced List									(0C00)
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	1	1	0	1	1										
n															
Reference to "A"															

Location "A"

List of n characters

3-37

3-26.   DF Field (Bits 6 and 7)

The Data Format field (DF), contained in bits 6 and 7 of the TEXT instruction, has the following values:

00	DFWD	Full word
01	DFBY	Byte
10	DF7B	7-bit format
11	DFRF	Refr

DFWD   (Full word) Character information is taken from the first byte of each list word.   The code "9C" will cause the data generation for this list to cease if LF Field uses terminate mode.   (Note: A 9C should not be used under count control.)   The order of bytes in a word corresponds to the assembler convention of the computer, defined by the BYTE 12 condition (BYTESWAP signal of GP Bus).

DFWD   Full Word   (0000)

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	1	1	0			0	0								
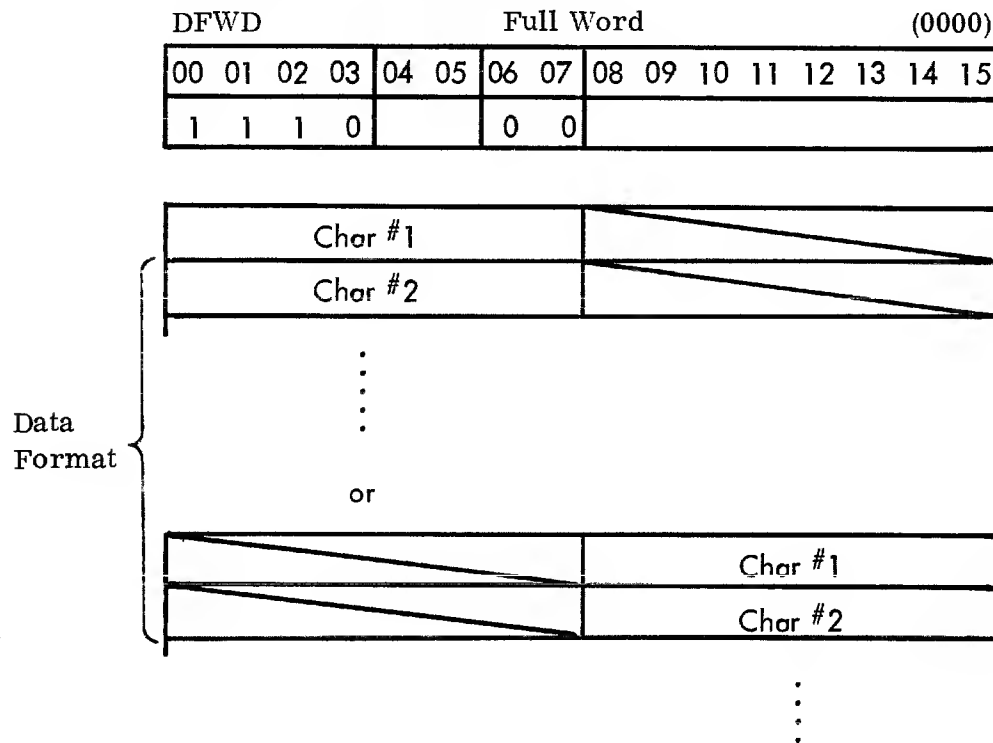
DFBY   (Byte) Character information is taken from successive bytes in the list, packed as two 8-bit bytes per word. This is the normal packing for text characters. The code "9C" will cause the data generation for this list to to cease if LF Field uses terminate mode. (Note: A 9C should not be used under count control.) The order of bytes in a word corresponds to the assembler convention of the computer, defined by the BYTE 12 condition (BYTESWAP signal of GP Bus).

DFBY   Byte   (0100)

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	1	1	0			0	1								

Data Format

Char	Char
Char	Char

DF7B   (7 bit) Each character value generated by the "Byte" processing above has its leftmost (8th) bit replaced by zero; thus passing only the 7-bit ASCII code after stripping any parity. The code "9C" will cause the data generation for this list to cease if LF Field uses terminate mode. (Note: A 9C should not be used under count control.). The leftmost bit (8th) of all CNTRL codes (except codes 30 through 0D) is set to "1". The order of bytes in a word corresponds to the assembler convention of the computer, defined by the BYTE 12 condition (BYTESWAP signal of GP Bus).

DF7B				7-Bit			(0200)								
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	1	1	0			1	0								

Data Format
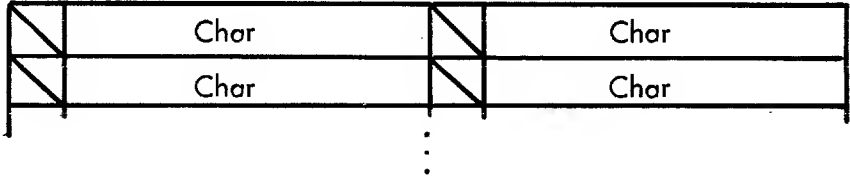
DFRF   (Refr) Each list word is used as a "Refr" to locate a value. Each value is considered a full word and character processing and termination is the same as under "Full Word" above.

DFRF   Reference   (0300)

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	1	1	0			1	1								

Data Format

Refr to Char #1
Refr to Char #2
. . .
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

NOTE: Due to look-ahead buffering of data in the DFRF (DF=11) format, the list must be followed by a NOOP instruction (word of zeros).

3-27.   PG Field (Bits 8 and 9)

The Page Control field (PG), comprised of bits 8 and 9 of the TEXT instruction, controls the location and margins for text data list processing. Format is shown below.

PG Field

00	PGNC	Current page
01	PGBM	Text block
10	PG00	Absolute page
11	PGXY	Positioned page

CURRENT PAGE TOP MARGIN

CURRENT Y POSITION

CURRENT PAGE LEFT MARGIN   CURRENT X POSITION

This is line one of text.
Carriage return places line two here.
Subsequent carriage returns will have a
similar
effect upon the current page, advancing
to the
next line, positioning the first character at
the left margin as shown.
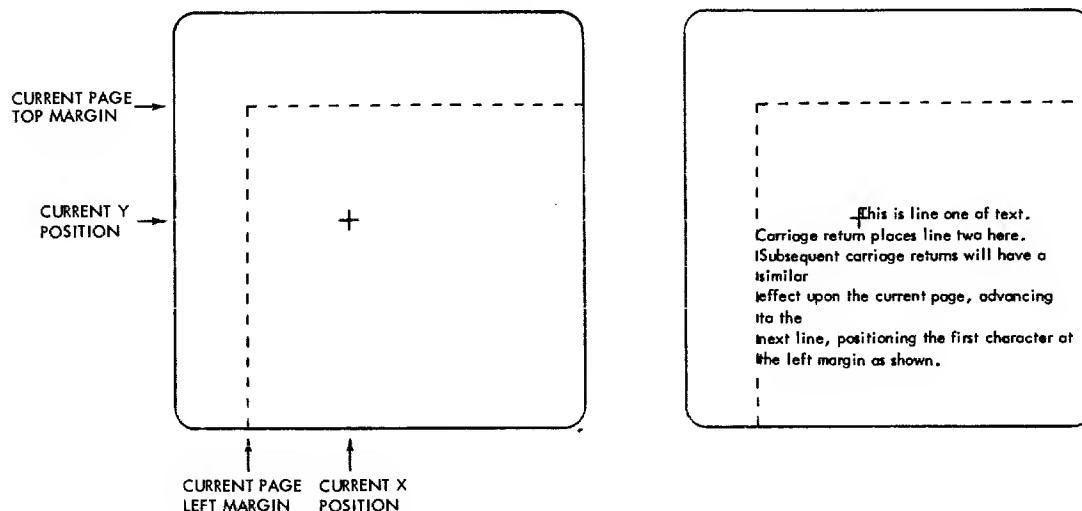
Figure 3-6. Current Page

PGNC   (Current Page) The text data is generated starting at the current co-ordinate position within the existing page margin registers (see Figure 3-6.

PGNC					Current Page							(0000)				
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	1	1	0					0	0						

CURRENT PAGE TOP MARGIN

CURRENT Y POSITION

Line one begins here.
Current X and Y
positions define new
margins. First
character is within
margins, down one-half
line feed.

CURRENT PAGE LEFT MARGIN
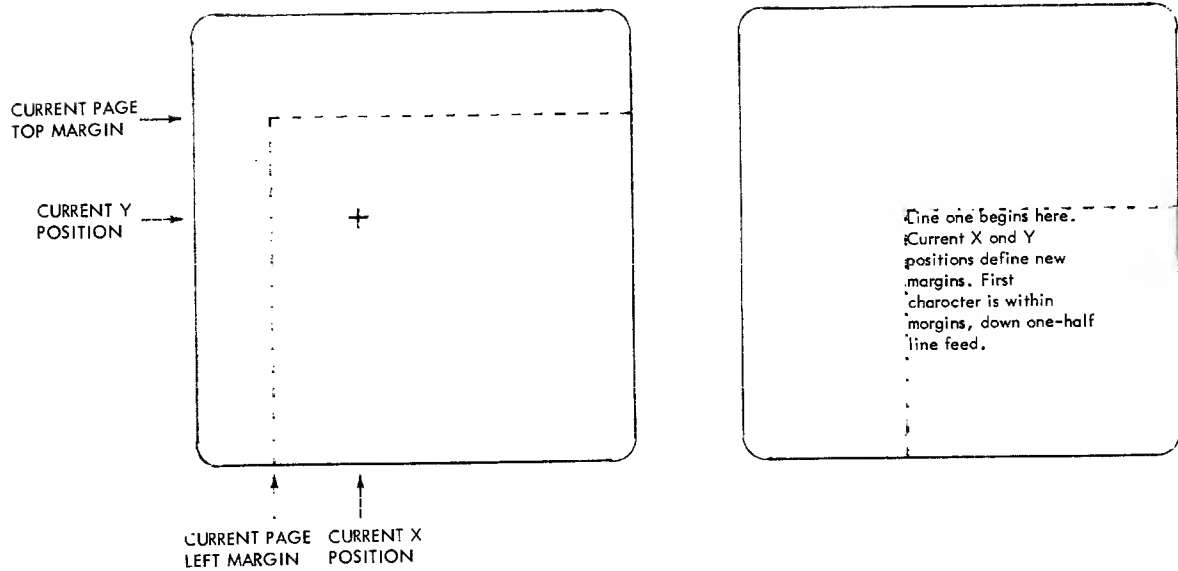
CURRENT X POSITION

Figure 3-7. Text Block

PGBM   (Text Block) The text data is generated starting at the current coordinate position; but first the X and Y register contents are copied in the page-left and page-top margin registers. This causes text lines to be written on a page whose upper left corner is positioned at the current coordinate location (see Figure 3-7).

PGBM   Text Block   (0040)

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	1	1	0					0	1						

MAXIMUM +Y
DEFLECTION

Line one of the text begins here. The first
character is automatically placed in the
position shown, very close to the maximum +Y
and maximum -X location.

CURRENT PAGE
TOP MARGIN

CURRENT Y
POSITION

CURRENT PAGE   CURRENT X
LEFT MARGIN   POSITION

MAXIMUM -X
DEFLECTION
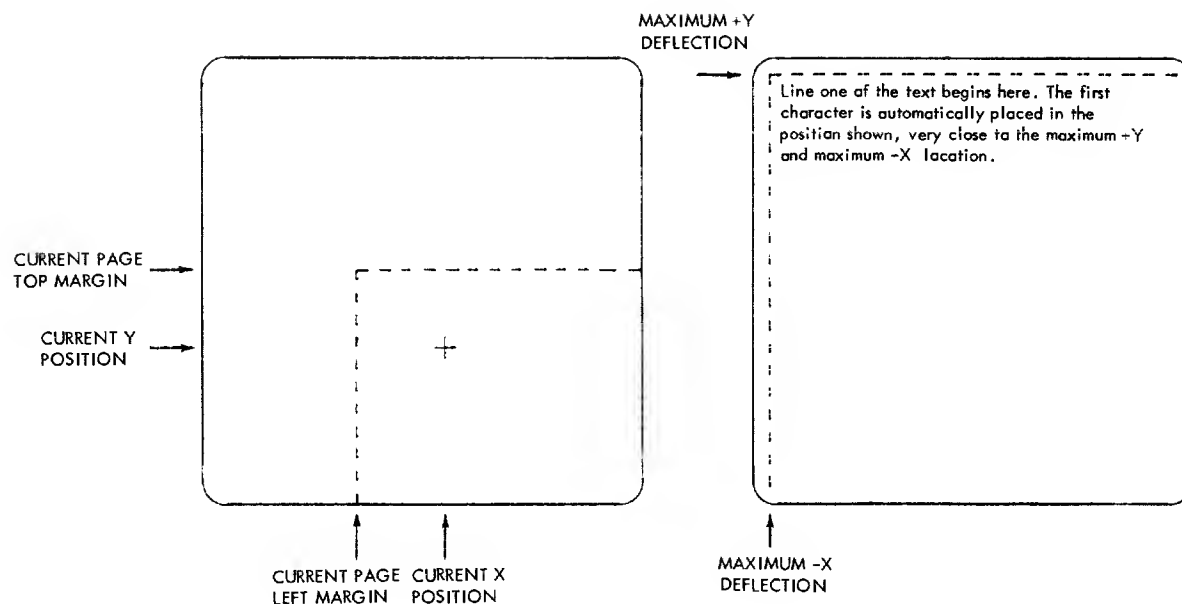
Figure 3-8. Absolute Page

PG00   (Absolute Page) Minimum and maximum values are first loaded into the
X and Y coordinate registers and into the page-left and page-top margin
registers respectively. The text data is then processed to generate
lines over a page positioned to fill the current X-Y plane at $Z = 0$ (see
Figure 3-8.

PG00				Absolute Page						(0080)					
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	1	1	0					1	0						
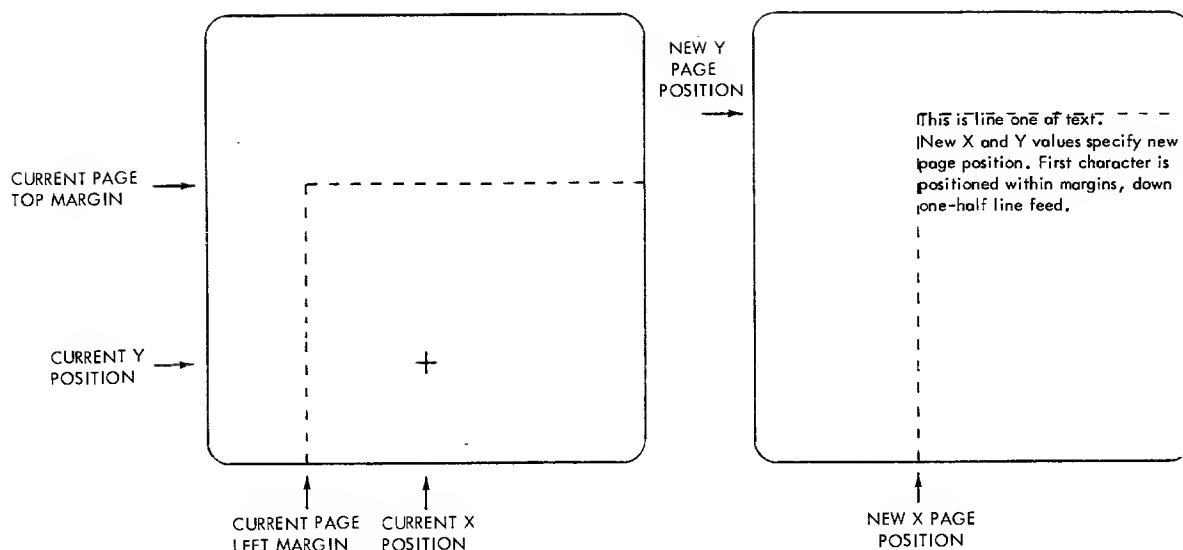
Figure 3-9. Positioned Page

PGXY   (Positioned Page) The first two items following the TEXT instruction word are taken as Refrs to X and Y page positioning values to be accessed and loaded into the X-Y coordinate registers and page-left/top margin registers prior to processing the text data list as per the LF and DF fields. If the X-Y Refrs are immediate, their values are multiplied by four (see Figure 3-9).

PGXY   Positioned Page   (00C0)

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	1	1	0					1	1						
New X Page Position																
New Y Page Position																

Three typical examples of the Positioned Page format are illustrated in Figure 3-10. Values for X and Y immediately follow the instruction word. Then, depending upon the condition of the LF field, information following the y value can be one of three possible combinations as shown.

TEXT   LFIT PGXY

00 01 02 03	04 05	06 07	08 09	10 11 12 13 14 15
1 1 1 0	0 0		1 1	
X			
Y			
Text Data			

TEXT   LFIC PGXY

00 01 02 03	04 05	06 07	08 09	10 11 12 13 14 15
1 1 1 0	1 0		1 1	
X			
Y			
Count			
Text Data			

TEXT   LFRC PGXY

00 01 02 03	04 05	06 07	08 09	10 11 12 13 14 15
1 1 1 0	1 1		1 1	
X			
Y			
Count			
Refr to Text Data			

Figure 3-10. Typical Formats for Positioned Page

3-28.   ROT Field (Bits 10 and 11)

The Rotation field (ROT), comprised of bits 10 and 11 of the TEXT instruction has
the following values:

ROT Field

00	RONC	(No change) Specifies no change in symbol orientation.
01	RO00	(Reset) Reset to zero-degree rotation.
10	ROPK	(Packed Refr-ed Word) Load rotation field into TXCT from the corresponding positions in the packed-type Refr-ed word as shown in Figure 3-11.
11	RORF	(Right-justified Refr-ed Word) Load rotation field into TXCT with bits positioned and extracted from the right-justified value in the corresponding Refr-ed word as shown in Figure 3-12.
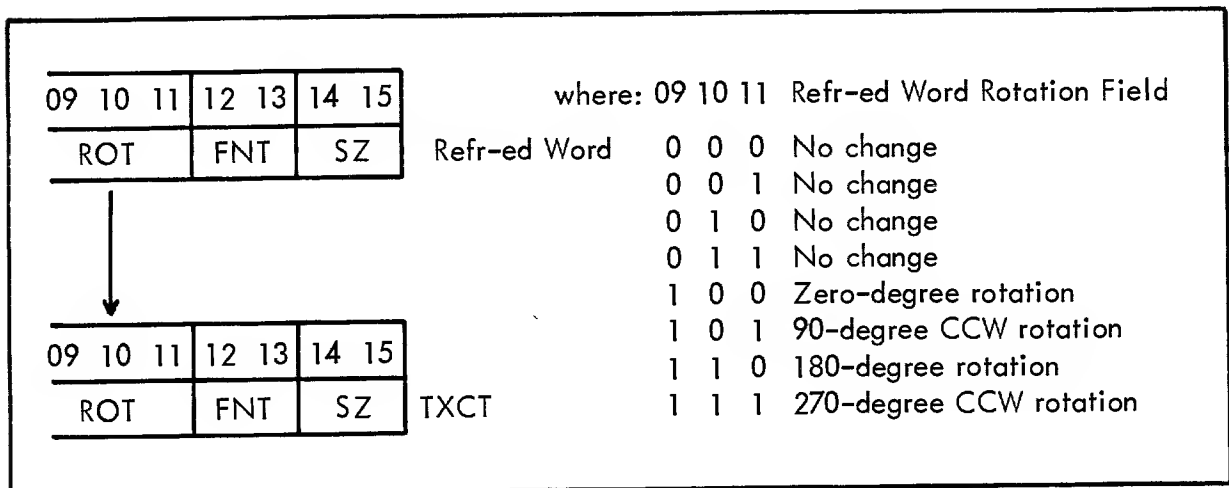
Figure 3-11. Rotation Field: Packed Refr-ed Word, ROPK
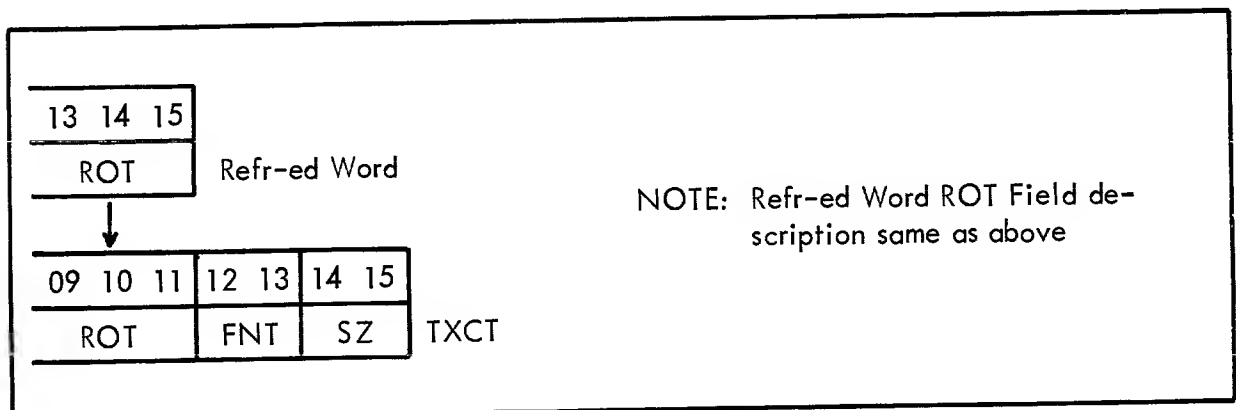
Figure 3-12. Rotation Field: Right-Justified Refr-ed Word, RORF

3-29.   <u>FNT</u> Field (Bits 12 and 13)

The Font field (FNT), comprised of bits 12 and 13 of the TEXT instruction, has the following values:

FNT Field

00	FNNC	(No change) Specifies no change in font.
01	FN00	(Reset) Reset font to normal (non-slanted symbols).
10	FNPK	(Packed Refr-ed Word) Load font field in TXCT from the corresponding positions in the packed-type Refr-ed word as shown in Figure 3-13.
11	FNRF	(Right-justified Refr-ed Word) Load font field into TXCT with bits positioned and extracted from the right-justified value in the corresponding Refr-ed word as shown in Figure 3-14.
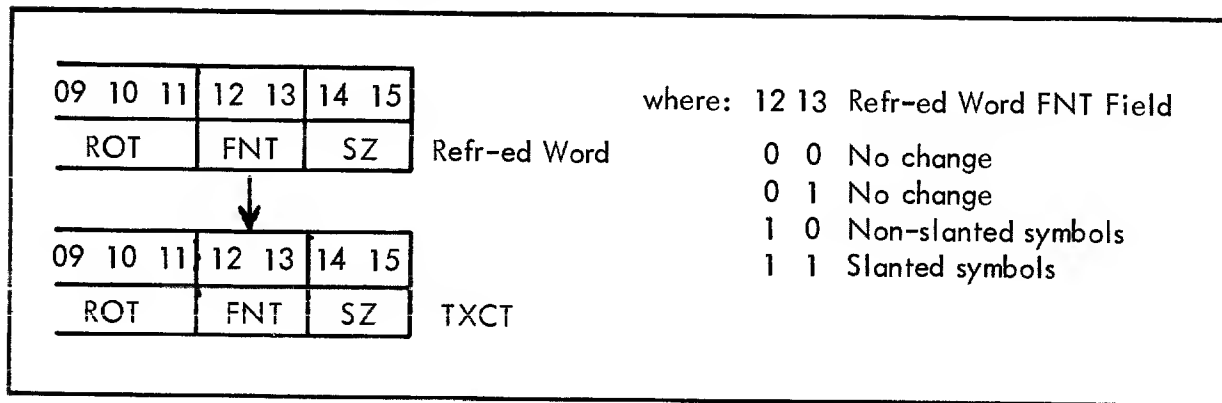
Figure 3-13.   Font Field: Packed Refr-ed Word, FNPK
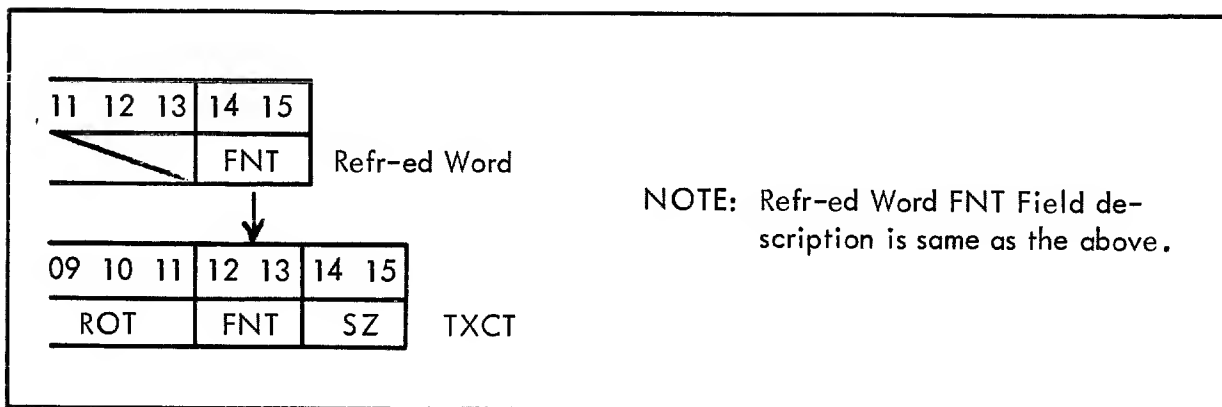
Figure 3-14.   Font Field: Right-Justified Refr-ed Word, FNRF

3-30. SZ Field (Bits 14 and 15)

The Size field (SZ), comprised of bits 14 and 15 of the TEXT instruction, has the following values:

SZ Field

00	SZNC	(No Change) Specifies no change in character size.
01	SZ80	(Default Size) Reset to standard 80 characters/line (Size #2)
10	SZPK	(Packed Refr-ed Word) Load size field into TXCT from the corresponding positions in the packed-type Refr-ed word as shown in Figure 3-15.
11	SZRF	(Right-justified Refr-ed Word) Load size field into TXCT with bits positioned and extracted from the right-justified value in the corresponding Refr-ed word as shown in Figure 3-16.
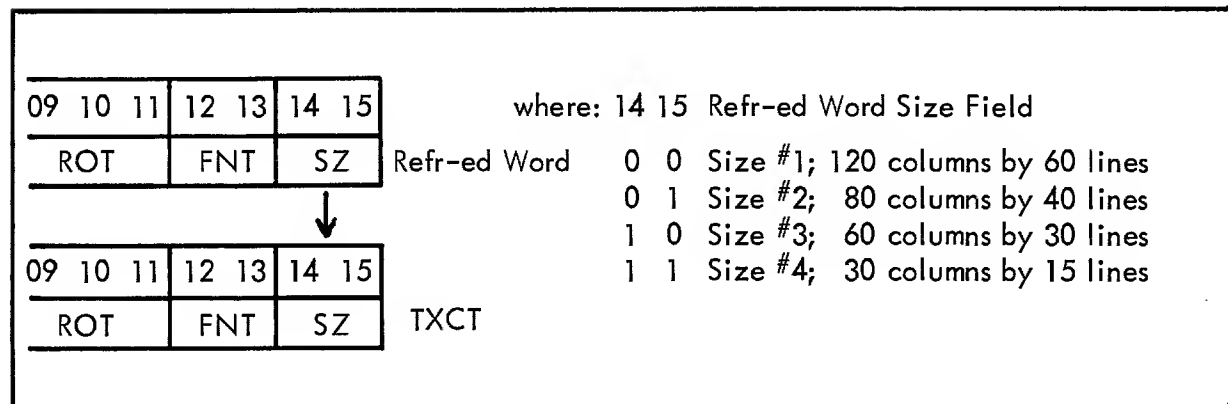
Figure 3-15. SZ Field: Packed Refr-ed Word, SZPK
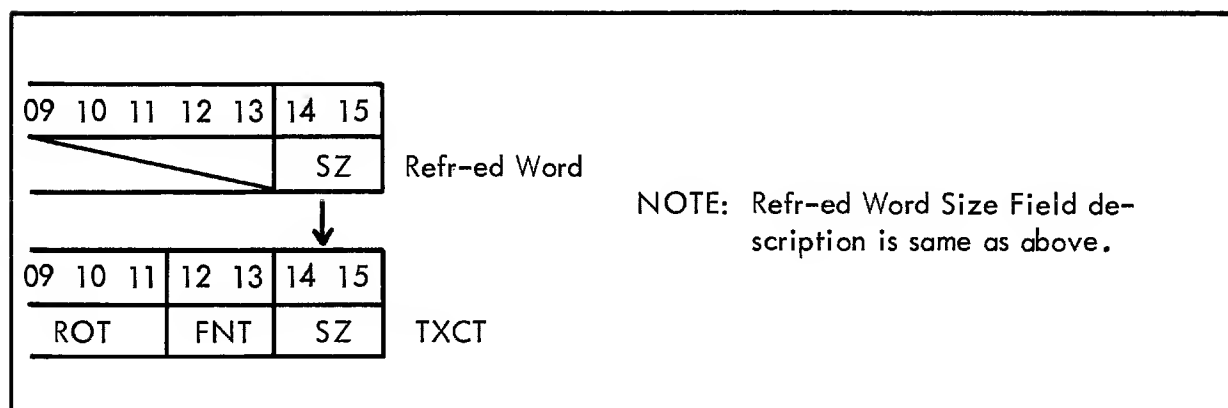
Figure 3-16. SZ Field: Right-Justified Refr-ed Word, SZRF

3-31 SINGLE ELEMENT INSTRUCTIONS (F000, F800)

The single element instructions create the graphic data to draw circles, arcs, rectangles, or cubic curve elements. Depending on the state of bit 04, the instruction may have one of two forms (see also Table 3-1):

a)   If bit 04 (M) is a one, arguments consisting of immediate values are left-shifted two bits (magnified 4 times). Referenced values are left unaltered.

b)   If bit 04 is a zero, no magnification is applied and the arguments are used, unalterd in magnitude, to create the graphic data.

Single Element Instructions (F000, F800)

00 01 02 03	04	05 06 07 08 09 10 11	12 13 14 15
1 1 1 1	M	0 0 0 0 0 0 0	OP
Refrs			

The OP field, bits 12 through 15, are decoded to provide the operations illustrated in Table 3-4. Descriptions of the corresponding operations are provided in the following paragraphs.

Table 3-4. Single Element Operations

OP CODE (bit 04 = 0)	Operation	OP Code (bit 04 = 1)	Operation
F000	CIRCLE	F800	CIRCL4
F001	CCWARC	F801	CCARC4
F002	CWARC	F802	CWARC4
F003	RECT	F803	RECT4
F004	CUBIC	F804	CUBIC4

3-32. Circle

Two of the single element operations described in Table 3-4 involve complete circles. The first Refrs following the instruction define the centerpoint X and Y coordinates of the circle. The third Refr defines the radius of the circle (from centerpoint in a +X direction) and is used to establish the start and endpoint of a circle drawn counterclockwise in the current Z plane as shown in Figure 3-17.
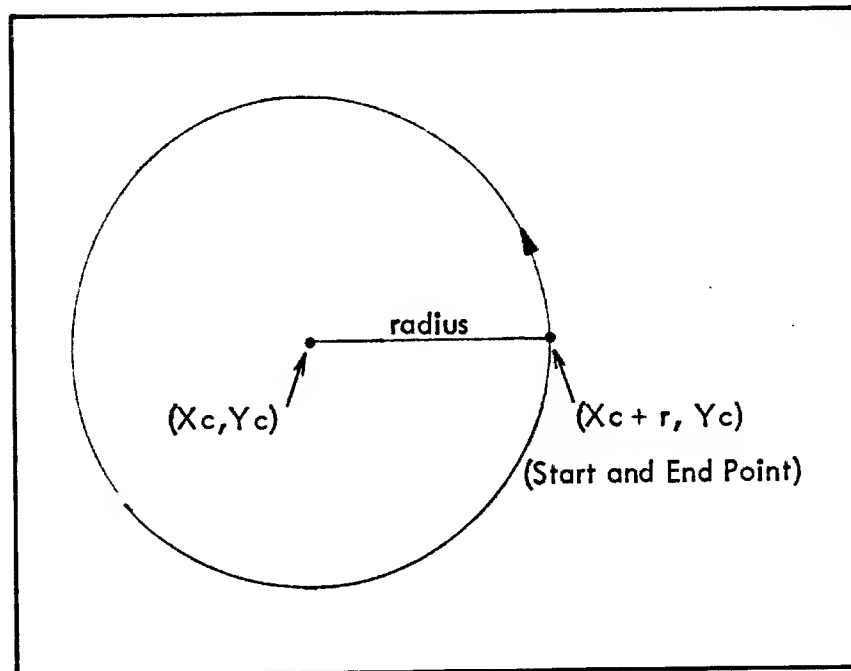
Figure 3-17. Coordinates For Circle Draw

CIRCLE   Single Element Circle   (F000)

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	1	1	1	0								0	0	0	0
Ref Xc															
Ref Yc															
Ref Radius															

3-33.   Arc

Four single element instructions generate arcs, two in a clockwise direction, two in
a counter-clockwise direction. These instructions are briefly described below.

3-34.   Counter-Clockwise Arc (CCWARC)

The next four Refrs define the X and Y coordinates of the centerpoint and endpoint of
an arc drawn counterclockwise from the current XYZ coordinates in the current Z
plane as follows:

Refr #1 = X1 = arc centerpoint X coordinate
Refr #2 = Y1 = arc centerpoint Y coordinate
Refr #3 = X2 = arc endpoint X coordinate
Refr #4 = Y2 = arc endpoint Y coordinate

NOTE: The distance between the arc center and endpoint coordinates must equal
the distance between the initial X/Y coordinate point and the arc centerpoint.

CCWARC					Single Element CCW Arc							(F001)			
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	1	1	1	0								0	0	0	1
Ref X Centerpoint															
Ref Y Centerpoint															
Ref X Endpoint															
Ref Y Endpoint															

Current point   ╋ Center point
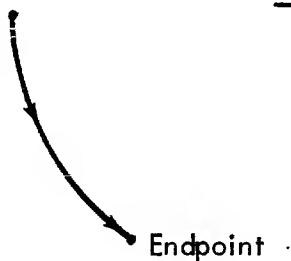
Endpoint

3-35.   Clockwise Arc (CWARC)

The next four Refrs define the X and Y coordinates of the centerpoint and endpoint of an arc drawn clockwise from the current XYZ coordinates in the current Z plane. The four Refrs hold the same coordinates as shown above for a CCW arc.

CWARC   Single Element CWARC   (F002)

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	1	1	1	0								0	0	1	0
Ref X Centerpoint															
Ref Y Centerpoint															
Ref X Endpoint															
Ref Y Endpoint															

3-36.   Rectangle

Two single element operations involve rectangles. The two Refrs following the instruction word (first X, then Y) define the coordinate position diagonally opposite the current XY coordinates. Figure 3-18 illustrates this scheme.
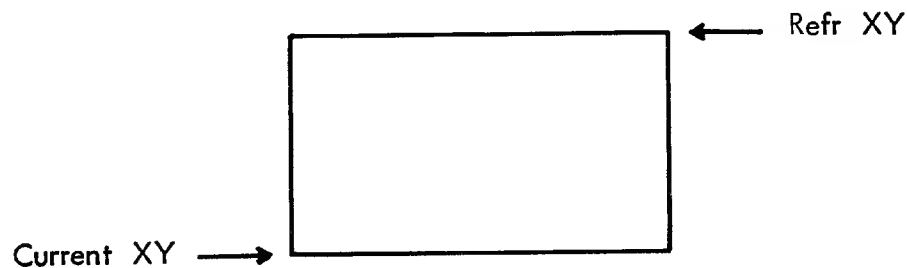
Figure 3-18.   Rectangle

RECT   Single Element Rectangle   (F003)

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	1	1	1	0								0	0	1	1
Ref X															
Ref Y															

3-37. Cubic

The cubic curve segment can be drawn by executing a CUBIC instruction (see Figure 3-19). This curve starts at the current position with a specified slope, quantity of segments, and endpoint. The seven Refrs following the instruction word are defined below. Figure 3-20 shows a typical curve.

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
1	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0
DELTA															
$\pm$	X														
$\pm$	Y														
$\pm$	Z														
$\pm$	DLTX														
$\pm$	DLTY														
$\pm$	DLTZ														

Figure 3-19. Cubic Instruction Format

DELTA = 1/N

    N = Number of points from T_0 to T_1

    X = X coordinate endpoint

    Y = Y coordinate endpoint

    Z = Z coordinate endpoint

DLTX = X slope at endpoint (left in DLTX register)

DLTY = Y slope at endpoint (left in DLTY register)

DLTZ = Z slope at endpoint (left in DLTZ register)
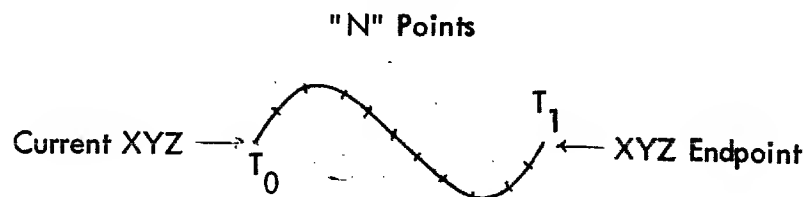
Figure 3-20. Typical Cubic

NOTE: Also, DLT X, DLT Y, and DLT Z registers must hold current (startpoint) slope values $\dfrac{\partial X(t)}{\partial t} \Big|\ t = 0$, etc.

3-38.   GPU ARGUMENT ADDRESSING

Most instructions require one or more values to specify their operation (i.e., coordinates for a line generation or angle for a rotation operator). These values may be provided in either of two ways:

   a) As immediate constants stored with each instruction, or

   b) As addressed references to variable values.

The above two operations are available to instructions which process a list or table of (possibly packed) values. The table may follow the instruction or, instead, an addressed reference which locates the table may be given.

3-39.   GPU REFERENCE ADDRESSING FORMATS

In all instruction descriptions, wherever a parameter may be specified by a general address reference ("Refr"), any of the addressing options illustrated in Table 3-5 may be coded.

Table 3-5. "Refr" Addressing Formats

00	01	02	03	04	05	06	07	08 09 10 11 12 13 14 15	DESCRIPTION
0	0	0						+ DIRECT VALUE	IMMEDIATE POSITIVE
0	0	1	0	0	0			DEVICE	DEVICE
0	0	1	0	1	IND		0	STACK INDEX	STACK – TEMPORARIES
0	0	1	0	1	IND		1	STACK INDEX	STACK – ARGUMENTS
0	0	1	1	0	IND		0	REGISTER	REGISTER
0	0	1	1	0	IND		1	REGISTER	REGISTER INCREMENTED
0	0	1	1	1	IND		1	REGISTER	REGISTER DECREMENTED
0	1	IND		OBJECT LOCAL STORAGE INDEX					LOCAL
1	0	IND		OBJECT LINK STORAGE INDEX					EXT. INDEXED VALUE
1	1	0	R	OBJECT LINK STORAGE INDEX					EXTERNAL VALUE
1	1	1		– DIRECT VALUE					IMMEDIATE NEGATIVE

Descriptions of the "Refr" addressing formats in Table 3-5 follows:

IMMEDIATE POSITIVE:	Value is a positive 13-bit value.
DEVICE:	Value is an I/O register address (see Appendix D).
STACK-TEMPORARIES:	Value is in the current level's (SA based) stack at location SA+ index.
STACK-ARGUMENTS:	Value is in the caller's stack at location (prev SA) + index
REGISTER:	Register contents.
REGISTER INCREMENTED:	Register contents after incrementing by one word-address count.
REGISTER DECREMENTED:	Register contents after decrementing by one word-address count.
LOCAL:	Value in object's local/own storage.
EXTERNAL INDEXED VALUE:	Next word Refrs index to external value.
EXTERNAL VALUE:	Directory (+link) holds address.
IMMEDIATE NEGATIVE:	Value is a negative 13-bit value.

Table 3-6 illustrates how the GPU handles the various Refr formats above and identifies the contents of addressed locations specified by (IND). Reference Figure 4-22.

Table 3-6. Coding For Refr Addressing

	Octal Code	Decimal Code	Hex Code	Mnemonic	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	Description
REF	00000	0000	0000	IMD+	0	0	0								–DIRECT VALUE						IMMEDIATE POSITIVE
	20000	8192	2000	DEV	0	0	1	0	0	0					I/O DEVICE ADDRESS						DEVICE
	24000	10240	2800	TMP	0	0	1	0	1	IND		0			STACK INDEX						TEMPS IN LOCAL STACK
	24400	10496	2900	ARG	0	0	1	0	1	IND		1			STACK INDEX						ARGS IN-CALLER'S STACK
	30000	12288	3000	REG	0	0	1	1	0	IND		0			REGISTER						REGISTER
	30400	12544	3100	RGI	0	0	1	1	0	IND		1			REGISTER						REGISTER INCREMENTED
	34400	14592	3900	RGD	0	0	1	1	1	IND		1			REGISTER						REGISTER DECREMENTED
	40000	16384	4000	LOC	0	1	IND			OBJECT LOCAL STORAGE INDEX											LOCAL
	100000	32768	8000	EXI	1	0	IND			OBJECT LINK STORAGE INDEX											EXTERNAL INDEXED VALUE
	140000	49152	C000	FXV	1	1	0	R		OBJECT LINK STORAGE INDEX											EXTERNAL VALUE
	160000	57344	E000	IMD-	1	1	1								-DIRECT VALUE						IMMEDIATE NEGATIVE
IND (STK/REG)	00000	0000	0000							0	0				REGISTER						DIRECT VALUE
	1000	512	0200	REF						0	1				REGISTER						NEW REFR (CANNOT BE TO EXI)
	2000	1024	0400	INW						1	0				REGISTER						WORD ADDR OF VALUE
	3000	1536	0600	INB						1	1				REGISTER						BYTE ADDR OF VALUE
IND (MEM)	00000	0000	0000					0	0	OBJECT LOCAL/OWN STORAGE INDEX											DIRECT VALUE
	10000	4096	1000	REF				0	1	OBJECT LOCAL/OWN STORAGE INDEX											NEW REFR (CANNOT BE TO EXI)
	20000	8192	2000	INW				1	0	OBJECT LOCAL/OWN STORAGE INDEX											WORD ADDR OF VALUE
	30000	12288	3000	INB				1	1	OBJECT LOCAL/OWN STORAGE INDEX											BYTE ADDR OF VALUE
R Field	00000	0000	0000						0	OBJECT LINK STORAGE INDEX											DIRECT VALUE
	10000	4096	1000	INW					1	OBJECT LINK STORAGE INDEX											WORD ADDR

3-40. GPU REGISTERS

Table 3-7 provides a list of GPU registers (in most cases microprocessor memory locations) which are accessible to the program. All registers, with exception of the STAT register, are addressable using programmed I/O (the STAT register can be read only). Paragraphs 3-41 through 3-53 provide descriptions of the contents and usage of each register.

Table 3-7. GPU Registers

HEX	DEC	OCT	REGISTER	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	USAGE
07	07	07	CMD.	NPIC	GO	PIC HE	SAM FRM	CPU STK	DEVICES STA#		RBU/DCU STA#		DCU IN	BUFF MODE		CLP TXT	CLP VEC			COMMAND
08	08	10	CTL.	ARB HE	BRK HE		SPIC HE	ELM HE	WRD HE	EDT HE	ERR HE	HIT HE	SEL HE		REG E	SEL E	HIT E		HINT E	CONTROL
09	09	11	STAT.	EDIT NSRT											STATE CODE					STATUS
00	00	00	DIR.	DIRECTORY ADDRESS																ADDRESS RELATED REGISTERS
01	01	01	STB.	CPU STACK EXTENSION (LOW ADDRESS)																
02	02	02	SLM.	CPU STACK EXTENSION (HI ADDRESS +1)																
03	03	03	OBA.	CURRENT OBJECT ADDRESS																
04	04	04	IA.	CURRENT INSTRUCTION ADDRESS																
0A	10	12	PBO.	OBJ# (DIR INDEX) OF BASE PICTURE																
0B	11	13	IR.	CURRENT INSTRUCTION IMAGE																
0C	12	14	OBN.	CURRENT OBJECT NUMBER																
0D	13	15	STK.	CURRENT STACK TOP INDEX																
0E	14	16	SA.	CURRENT STACK LEVEL INDEX																
0F	15	17	GP1.	GENERAL PURPOSE REGISTER #1																GP REGISTERS
10	16	20	GP2.	GENERAL PURPOSE REGISTER #2																
11	17	21	GP3.	GENERAL PURPOSE REGISTER #3																
12	18	22	GP4.	GENERAL PURPOSE REGISTER #4																
13	19	23	PWC.	PICTURE WORD COUNT																PICTURE RELATED REGISTERS
14	20	24	PS.	0	PICTURE SCALE															
15	21	25	PSI.	0	PICTURE INTENSITY DEPTH CUEING															
16	22	26	PDX.	±	PICTURE X DISPLACEMENT															
17	23	27	PDY.	±	PICTURE Y DISPLACEMENT															
18	24	30	PDZ.	0	WINDOW PERSPECTIVE DEPTH CUEING (Z VIEWPOINT)$^{-1}$															
19	25	31	PDI.	0	MAXIMUM PICTURE INTENSITY															
1A	26	32	WCX.	±	WINDOW X CENTER POINT															
1B	27	33	WCY.	±	WINDOW Y CENTER POINT															
1C	28	34	WNZ.	±	3D WINDOW Z NEAR CUTOFF PLANE															
1D	29	35	WSX.	0	WINDOW HORIZONTAL SIZE															
1E	30	36	WSY.	0	WINDOW VERTICAL SIZE															
1F	31	37	WSZ.	0	3D WINDOW Z DEPTH															
20	32	40	DS.	0	INPUT DATA SCALE															

Table 3-7. GPU Registers (Continued)

HEX	DEC	OCT	REGISTER	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	USAGE
21	33	41	OS.	0	OBJECT SCALE															OBJECT RELATED REGISTERS
22	34	42	ODX.	±	OBJECT X DISPLACEMENT															
23	35	43	ODY.	±	OBJECT Y DISPLACEMENT															
24	36	44	ODZ.	±	OBJECT Z DISPLACEMENT															
25	37	45	RX.	±	OBJECT X ROTATION															
26	38	46	RY.	±	OBJECT Y ROTATION															
27	39	47	RZ.	±	OBJECT Z ROTATION															
05	05	05	DA.	CURRENT DATA ADDRESS																DATA TABLE REGISTERS
28	40	50	ELN.	ELEMENT NUMBER																
29	41	51	LNCT.				3DI	BLINK		COLOR		LINE TYPE			VECTOR MODE					
2A	42	52	TXCT.				BLINK		COLOR		ORIENTATION			FONT			SIZE			
2B	43	53	COLR.	COLOR																
2C	44	54	PGT.	±	PAGE TOP Y COORDINATE															
2D	45	55	PGL.	±	PAGE LEFT X COORDINATE															
2E	46	56	X.	±	CURRENT X COORDINATE															ELEMENT RELATED REGISTERS
2F	47	57	Y.	±	CURRENT Y COORDINATE															
30	48	60	Z.	±	CURRENT Z COORDINATE															
31	49	61	INTN.	±	CURRENT INTENSITY LEVEL															
32	50	62	DLTX.	±	X INCREMENT															
33	51	63	DLTY.	±	Y INCREMENT															
34	52	64	DLTZ.	±	Z INCREMENT															
35	53	65	DLTI.	±	INTENSITY INCREMENT															
36	54	66	SELWC.	SELECT PICTURE WORD COUNT																SELECT REGISTERS
37	55	67	SELCT.	SEL MODE															ODD BYTE	
38	56	70	HITDEV.	SEL/HIT DEVICE																
39	57	71	PIKX.	±	PICK WINDOW X CENTER															PICK OPTION REGISTERS
3A	58	72	PIKY.	±	PICK WINDOW Y CENTER															
3B	59	73	PIKS.	0	PICK WINDOW SIZE (OPTIONALLY X ONLY)															
3C	60	74	PIKSY.	0	PICK WINDOW Y SIZE (OPTIONAL)															
3D	61	75	HN.													HIT COUNT				HIT REGISTERS
3E	62	76	HITCT.	HIT MODE			DET MODE					HIT UNIT								
3F	63	77	HITWC.	HIT WORD COUNT																
40	64	100	HITEC.	HIT ELEMENT COUNT																
06	06	06	EA.	EDIT ADDRESS																EDIT REGISTERS
41	65	101	EPWC.	EDIT PICTURE WORD COUNT																
42	66	102	ESC.	EDIT SKIP COUNT																
43	67	103	EIC.	EDIT INSERT COUNT																
7B	123	173	LOUT. *											ZL	ZH	YL	YH	XL	XH	
FF	255	377	RWC.	RBU STORE INDEX																

*CLIPPER VIOLATION FLAGS

3-41.   COMMAND REGISTER (CMD), Hex Address 007

The Command Register (CMD), a user addressable register contained in address 007
of the RAM in the GPU, functions to store data from the host computer. This infor-
mation is subsequently utilized by the GPU microprogram to control specified opera-
tions. Bit assignments for the Command Register are listed in Table 3-8.

Table 3-8.   Command Register Bit Assignments

Bits	Mnemonic	Description
0	NPIC	New picture bit. Functions to initialize the specified picture, object, transform, data table, element generation, edit, select, and hit operations.
1	GO	Specifies execution or suspension of GPU processing.
2	PICHE	Picture halt enable. 0 = continuous update processing. 1 = halts GPU processing at conclusion of specified picture to provide single-update pass. NOTE: Interrupt to CPU is generated if HINTE = 1 (in CTL Register) and PICHE = 1.
3	SAMFRM	Same frame. Permits output frame to be built from independent input pictures under current buffering or edit modes.
4	CPU STK	Force stack into CPU. Inhibits allocation of stack space to hardware RAM. This permits program access to the entire stack space.
5,6	DEVSTA	Device station number field. These bits are added to most significant two bits of input device addresses to allow same list to be assigned to any of four device groups.
7,8	RBU/DCU STA	RBU/DCU station number field. 00 = select RBU/DCU pair #1 01 = select RBU/DCU pair #2 10 = select RBU/DCU pair #3 11 = select RBU/DCU pair #4
9	DCUIN	Initialize DCU. Starts DCU, sets Frame Rate to 40 Hz, Frame Mode to ALL, and selects Monitor #1 at highest vector-draw rate.

Table 3-8. Command Register Bit Assignments (Continued)

Bits	Mnemonic	Description
10, 11	BUFMOD	Buffer mode field. 00 = single buffer mode. 01 = double buffer mode. 10 = edit insert only-buffer mode. 11 = edit insert to end of frame.
12	CLPTXT	Enables clipping when character text mode is chosen.
13	CLPVEC	Enables clipping when vector mode is chosen.

3-42. CONTROL REGISTER (CTL), Hex Address 008

The Control Register (CTL), a user addressable register contained in address 008 of the RAM in the GPU, functions to store data from the host computer. This information is used to perform the following: (a) determine whether the GPU will halt under certain known conditions, (b) control the Select, Hit, and Edit features, and (c) determine whether the host computer should be interrupted when a halt occurs. Hit assignments for the Control Register are listed in Table 3-9. Bits 0 through 9, inclusive, along with bit 2 of the Command Register (CMD), constitute the conditional halt control bits. Bits 12 through 13 inclusive, comprise the feature controls; bit 15 is an interrupt control.

Table 3-9. Control Register Bit Assignments

Bits	Mnemonic	Description
0	ARBHE	ARB instruction halt-enable bit. When set, GPU halts whenever ARB or ARBI instruction is received.
1	BRKHE	BRKL instruction halt-enable bit. When set, GPU halts whenever BRKL instruction is received.
3	SPICHE	Sub-picture halt-enable bit. When set, GPU halts when a CALLU or CALLC instruction is received.
4	ELMHE	Element halt-enable bit. When set, GPU halts for each element processed; (e.g., each vector, each character, or a LOAD, NEST instruction).

Table 3-9. Control Register Bit Assignments (Continued)

Bits	Mnemonic	Description
5	WRDHE	Input word halt-enable bit. When set, GPU halts after each input word is read.
6	EDTHE	Edit address halt-enable bit. When set, GPU halts when input fetch address matches address specified by the EA Register.
7	ERRHE	Error halt-enable bit. When set, GPU halts upon detection of any of the following: addressing error, invalid instruction, invalid argument, illegal register, graphic stack overflow, RBU full, transform out of range, or invalid directory.
8	HITHE	HIT halt-enable bit. When set, GPU halts when HITE bit (bit 13) is set and the refresh word count (RWC) matches the hit word count (HWC).
9	SELHE	SELECT halt-enable bit. When set, GPU halts when picture word count (PWC) matches the select word count (SELWC).
11	REGE	Permits read access to all GPU registers.
12	SELE	Select feature enable bit.
13	HITE	Hit feature enable bit.
15	HINTE	Interrupt-on-halt enable bit. When set, the GPU generates an interrupt to the host computer whenever the Halt State is entered. At the time of such a GPU halt, the STATUS Register is loaded with data relating to the cause of the Halt State.

3-43. STATUS REGISTER (STAT), Hex Address 009

The Status Register (STAT), a user addressable register contained in address 009 of the RAM in the GPU, stores status information which is accessible to the host computer. Bit assignments of the Status Register are listed in Table 3-10.

The conditions which set state codes 1 and 3 through 14 will also generate an interrupt to the CPU if the HINTE-bit in the CTL Register is set.

Table 3-10. Status Register Bit Assignments

Bits	Mnemonic	Name	Description
0	EDTNSRT	Edit Insert	Indicates that an edit-mode update is in progress.
11-15	STACD	State Code	Provides status. 00 = host CPU bus reset 01 = ready (GO-bit in CMD Register is 0) 02 = running 03 = CPU memory addressing error 04 = invalid graphic instruction 05 = invalid argument 06 = illegal register number or register or register set code (i.e., NEST, NEST1) 07 = graphic stack overflow 08 = RBU overflow 09 = transform out of range 0A = invalid PBO or Directory structure 0B = edit address halt (edit address matches IA or DA) 0C = hit halt 0D = select halt (SELWC matches PWC) 0E = halt instruction 0F = frame halt 10 = sub-picture halt 11 = BRKL instruction halt 12 = element halt 13 = word halt 14 = ARB instruction halt

3-44.   ADDRESS-RELATED REGISTERS (refer to Table 3-7)

DIR   A 16-bit register, maintained by the host computer, which contains the absolute address of the directory table in the host computer. NOTE: This register must be initialized by a programmed output prior to an NPIC command.

STB   A 16-bit register which contains the address of the stack base as defined by the host computer (lowest address of stack area). NOTE: This register must be initialized by a programmed output prior to an NPIC command.

SLM   A 16-bit register which contains the address of the stack limit as defined by the host computer [highest address (+1 word) of stack area]. NOTE: This register must be initialized by a programmed output prior to an NPIC command.

OBA   A 16-bit address register, maintained by the GPU, which contains the absolute address of the object currently being processed. The contents of this addressed loaction is the count of the number of words from the start of the object to the first instruction to be processed. It is also the Base for indexed access to links and locals located in the current object list. Derived from the contents of DIR and OBN as follows:

$$OBA = (DIR \text{ plus } OBN)$$

IA   A 16-bit address register, maintained by the GPU, which contains the absolute address of the instruction word in the object currently being used. The initial value of IA is the object address plus the contents of the word at the object address as follows:

$$IA \text{ initial} = OBA \text{ plus } (OBA)$$

PBO   A 16-bit register, maintained by the host computer, which holds the count of the number of words from the start of the directory to the entry containing the main picture base object address. NOTE: This register must be initialized by a programmed output prior to an NPIC command.

IR   A 16-bit register, maintained by the GPU, which contains a copy of the current GPU instruction.

OBN   A 16-bit register, maintained by the GPU, which contains the count of the number of words from the start of the directory to the object currently being processed. The OBN initial value is set by the GPU to the PBO number. Subsequent loading is from a CALL instruction to the next lower level and the RETURN instruction from the subroutine stack to the next higher level.

STK   A 16-bit register, maintained by the GPU, which contains the index of the current level top-of-stack.

SA   A 16-bit register, maintained by the GPU, which contains the index of the stack base for the current level. The first word of the current object's local stack is addressed by SA + 1.

3-45.   GENERAL PURPOSE REGISTERS (refer to Table 3-7)

Four 16-bit general purpose registers (GP1, GP2, GP3, and GP4) are available to the user for use in referencing addressing (direct, indirect, indexing, etc.) and temporary argument storage. The value held in GP1 controls the operation of the conditional CALL and RETURN instructions.

3-46.   PICTURE RELATED REGISTERS (refer to Table 3-7)

PWC   A 16-bit register, maintained by the GPU, which contains the count of the word from the host computer data base which will be processed next by the GPU.

PS   A 16-bit register, updated by the display list or by a programmed output from the host computer, which contains the scale value used to scale all transformed objects and elements to establish final picture size.

PSI   A 16-bit intensity scale register, updated by the display list or by a programmed output from the host computer, which specifies the amount of intensity depth cueing to be evident in the display. It controls intensity modulation to vary from uniform intensity at PSI = 0 to a maximum intensity variation from bright front to dim rear when PSI = full scale.

PDX   A 16-bit register, updated by the display list or by a programmed output from the host computer, which contains the picture X displacement value used to position the final 2-D projected view on the display screen.

PDY   A 16-bit register, updated by the display list or by a programmed output from the host computer, which contains the picture Y displacement value used to position the final 2-D projected view on the display screen.

PDZ   A 16-bit register, updated by the display list or by a programmed output from the host computer, which establishes the picture perspective Z viewpoint. The window X and Y extraction boundaries, and also the displayed data, will vary from parallel to 90° divergent as PDZ is varied from zero to maximum.

PDI   A 16-bit displacement register, updated by the display list or by a programmed output from the host computer, which establishes the maximum intensity of the brightest element in the display. All other elements are then displayed with proportional intensities.

WCX   A 16-bit window register, updated by the display list or by a programmed output from the host computer, which contains the X coordinate value of the center point of a portion of the data space which is to be extracted and presented for viewing in a window area specified by PDX, PDY and PS.

WCY   A 16-bit window register, updated by the display list or by a programmed output from the host computer, which contains the Y coordinate value of the centerpoint of a portion of the data space which is to be extracted and presented for viewing in window area specified by PDX, PDY and PS.

WNZ   A 16-bit window register, updated by the display list or by a programmed output from the host computer, which controls the value of the front (near) Z cutoff plane when 3-D clipping is used in the window feature.

WSX   A 16-bit window register, updated by the display list or by a programmed output from the host computer, which establishes the size, from the center point out to $\pm$ WSX horizontally, of a data portion to be extracted for viewing when using the window feature.

WSY   A 16-bit window register, updated by the display list or by a programmed output from the host computer, which establishes the size, from the centerpoint out to $\pm$ WSY vertically, of a data portion to be extracted for viewing when using the window feature.

WSZ   A 16-bit window register, updated by the display list or by a programmed output from the host computer, which establishes the depth, behind the near cutoff plane WNZ, of a data portion to be extracted for viewing when using the 3-D window option. This establishes the inner Z boundary at which the extracted data will be clipped.

DS   A 16-bit register, updated by the display list or by a programmed output from the host computer, which is used to scale all incoming, untransformed, coordinate data.

3-47.   OBJECT RELATED REGISTERS (refer to Table 3-7)

OS   A 16-bit register, updated by the display list or by a programmed output from the host computer, which contains the scale factor used for the object currently being processed.

ODX   A 16-bit register, updated by the display list or ba a programmed output from the host computer, which contains the X displacement value of the object currently being processed.

ODY   A 16-bit register, updated by the display list or by a programmed output from the host computer, which contains the Y displacement value of the object currently being processed.

ODZ   A 16-bit register, updated by the display list or by a programmed output from the host computer, which contains the Z displacement value of the object currently being processed.

RX   A 16-bit register, updated by the display list or by a programmed output from the host computer, which contains the current angle of X rotation of the object currently being processed (or the initial Z rotation in a NEST ZYZ operation).

RY   A 16-bit register, updated by the display list or by a programmed output from the host computer, which contains the current angle of Y rotation (performed after RX) of the object currently being processed.

RZ   A 16-bit register, updated by the display list or by a programmed output from the host computer, which contains the current angle of Z rotation (performed after RX and RY) of the object currently being processed.

3-48.   DATA TABLE REGISTERS (refer to Table 3-7)

DA   A 16-bit address register, maintained by the GPU, which contains the address of the data word currently being processed by the GPU.

ELN   A 16-bit number register, maintained by the GPU, which contains the element number (in current object) of the word currently being processed by the GPU.

LNCT A 16-bit control register, updated by the display list or by a programmed output from the host computer, which specifies the type of line the VGU is directed to draw. It contains the following fields: 3-DI, Blink, Color, Line Type, Incremental and Smoothed-Incremental Modes. See page 5-4 for descriptions of the Line Type fields. See page 5-11 for description of the Vector Mode field, and page 5-12 for description of the Blink field.

TXCT A 16-bit control register, updated by the display list or by a programmed output from the host computer, which specifies the Text Size, Orientation, Blink, and Font Type for character draws.

COLR A register, updated by the display list or by a programmed output from the host computer, which holds the color code for the MCU. This value is the initial or default color code when not overridden by LNCT or TXCT values.

PGT A 16-bit register, updated by the display list or by a programmed output from the host computer, which contains the Y coordinate value to establish the top of a page of text on the display.

PGL A 16-bit register, updated by the display list or by a programmed output from the host computer, which contains the X coordinate value to establish the left edge of a page of text on the display.

3-49. ELEMENT RELATED REGISTERS (refer to Table 3-7)

X A 16-bit register, updated by the display list or by a programmed output from the host computer, which contains the untransformed X coordinate value for element generation.

Y A 16-bit register, updated by the display list or by a programmed output from the host computer, which contains the untransformed Y coordinate value for element generation.

Z A 16-bit register, updated by the display list or by a programmed output from the host computer, which contains the untransformed Z coordinate value for element generation.

INTN A 16-bit intensity register, updated by the display list or by a programmed output from the host computer, which specifies intensity levels for displayed elements. When depth cueing is used, this value is combined with the Z axis value and PSI to provide intensity modulation.

DLTX   A 16-bit register, updated by the display list or by a programmed output from the host computer, which contains the untransformed X incremental data value to be added to any graph being generated or initial X slope for CUBIC operation.

DLTY   A 16-bit register, updated by the display list or by a programmed output from the host computer, which contains the untransformed Y incremental data value to be added to any graph being generated or initial Y slope for CUBIC operation.

DLTZ   A 16-bit coordinate register, updated by the display list or by a programmed output from the host computer, which contains the untransformed Z incremental data value to be added to any graph being generated or initial Z slope for CUBIC operation.

DLTI   A 16-bit coordinate register, updated by the display list or by a programmed output from the host computer, which contains the incremental intensity value to be added to the current element being generated.

3-50.   SELECT FEATURE REGISTERS (refer to Table 3-7)

SELWC A 16-bit control register, updated by the display list or by the host computer with the picture word count of any item selected by program for highlighting, or to be correlated to an Address, Object#, Element#, etc., by Select Halt.

SELCT A control register, updated by the display list or by the host computer, which contains the select code for the type of item-highlighting to be used.

HITDEV A device select register, loaded by the host computer, which contains the code for device selection. It permits use of a device-generated "Hit" (Pen or Pick) in place of an SELWC word count match for item highlighting.

3-51.   PICK OPTION REGISTERS (refer to Table 3-7)

PIKX   A 16-bit coordinate register, updated by the display list or by the host computer, which contains the X coordinate centerpoint of the Pick Device Window.

PIKY   A 16-bit coordinate register, updated by the display list or by the host computer, which contains the Y coordinate centerpoint of the Pick Device Window.

PIKS   A 16-bit Pick Device window size register, updated by the display list or by the host computer, whose contents specify the X and Y size of a Pick Device square window. Optionally, a Y size register may be added (PIKSY), in which case this register specifies only the X coordinate of window size.

PIKSY   A 16-bit optional Pick Device window register, updated by the display list or by the host computer, which specifies the Y size of the device window (see PIKS above).

3-52.   HIT FEATURE REGISTERS (refer to Table 3-7)

HN   A 4-bit number register, updated by the display list or by the host computer, which contains a count of the number of Pick Device window boundary crossings, or Light Pen detects, to accept before generating a "Hit." (Example: if HN = 0010, generate "Hit" on 2nd detect.)

HITCT   A control register, updated by the display list or by the host computer, which enables and specifies the Hit Mode and the Detect Mode (enables or disables counting of window boundary crossings or pen detects under switch control). The Unit field is set by the GPU during "Hit Halt" to identify which HIT DEVICE generated the "Hit" (Devices 1, 2, 3, and 4 set bits 8, 9, 10 and 11, respectively.

HITWC   A 16-bit picture word count register, maintained by the GPU, which holds the comput data base word count (PWC) of the chosen word which was "hit" by a device.

HITEC   A 16-bit element count register, maintained by the GPU, which contains the element count (in object) of the desired element which was "hit" by a device.

3-53.   EDIT FEATURE REGISTERS (refer to Table 3-7)

EA   A 16-bit edit address register, loaded by the host computer, which contains the address of a buffer in the host CPU containing an edit-insert sequence; or address to be correlated to a PWC by the Edit Address Halt. NOTE: When used to edit, this register must be initialized by programmed output prior to NPIC command.

EPWC   A 16-bit edit picture word count register, loaded by the host computer, which contains the picture word count (PWC) at which an edit sequence (insertion or deletion) is to be started. NOTE: This register must be initialized by programmed output prior to NPIC command.

ESC   A 16-bit edit word skip count register, loaded by the host computer, which
      contains the count of the number of words in the computer data base to be
      skipped. NOTE: This register must be initialized by programmed output
      prior to NPIC command.

EIC   A 16-bit edit insert count register, loaded by the host computer, which con-
      tains the count of the number of words in an edit sequence to be added to the
      displayed image (the length of the data list pointed at by EA). NOTE: This
      register must be initialized by programmed output prior to NPIC command.

3-54.   ROTATIONAL RESOLUTION

The user is not burdened with generating and specigying trigonometric functions for
rotational values; he need only specify the desired rotation angle in semi-circles
using up to 15 bits plus sign which provides a resolution of .0055° or 19.78 seconds
of arc as shown in Figure 3-21.
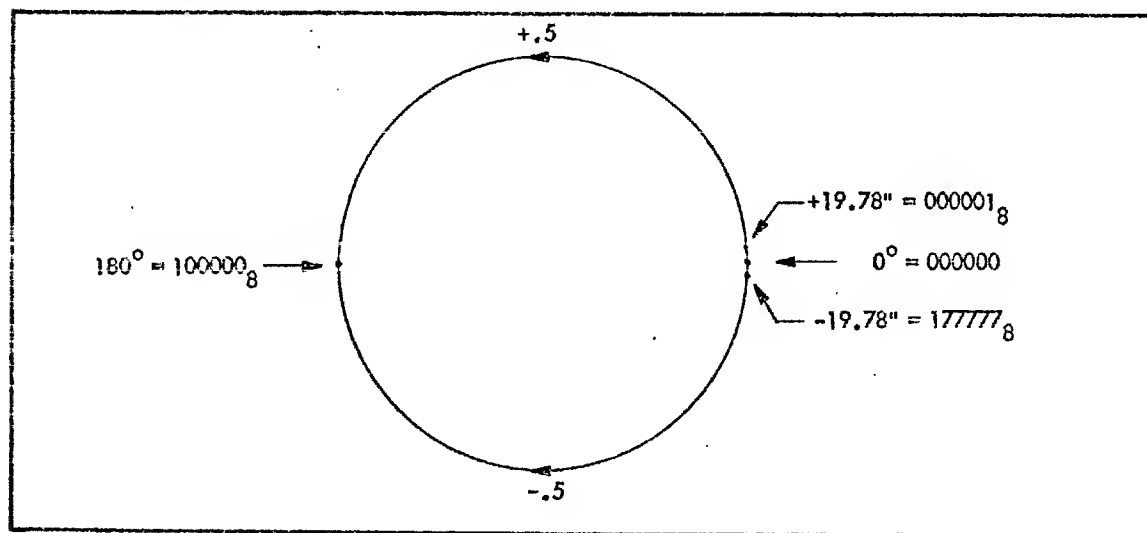
Figure 3-21.   Rotational Resolution

SECTION IV

SYSTEM PROGRAMMING

4-1.   INTRODUCTION

This section provides examples of how the user's graphic data may be used to generate display objects. In addition, enable and halt usage, the use of sub-object calls, transforms, windowing, addressing, stack and viewing parameters are also described.

4-2.   ENABLE AND HALT USAGE

Usually, the unique specification of all displayed items is done using the item's Picture Word Count (PWC). The PWC is used to communicate the unique identity of displayed data items between a program and a user/operator when using interaction-aid options.

If a program requires the PWC for a displayed item and already knows the memory location of the item, the program may acquire the PWC as follows: (a) the address of the item is loaded into the Edit Address (EA) register, (b) the GPU is run with the Edit Halt Enable bit ON (CTL [106]), and (c) the Edit Enable bit is OFF (CTL [14]). When the GPU halts with the Edit address Halt state code in the Status register, the GPU PWC register will hold the PWC of the addressed item.

The PWC of an item may be used to obtain any other information on its display usage. By loading the item's PWC into SELWC and running the GPU with Select Halt Enable bit ON (CTL [09]), when the GPU pauses, the item's picture, object, data and element processing information may be read from the GPU registers. The picture processing information would consist of the main object number, directory address, picture element count, word count, stack pointers, picture transform and clipping registers. The object information includes current object number, instruction word address, object base address, and all transformation registers (scale, displacement, rotation). The data list information includes the instruction and data-list addresses, element and word counts, line type, text type, text size, intensity, color, and page margin registers. The element information includes the instruction, character, coordinate, and increment registers whose parameters are used to generate visual elements. Other information is also available such as the current RBU address. When the GPU has halted, operation may be resumed by setting the GO bit (CMD [01]).

A momentary pause of the GPU operation may be monitored under program control by sensing the State Code field in the STATUS register or by an interrupt. If the "Interrupt-on-halt enable" bit (CTL [15]) is set, any enabled halt condition will not only suspend GPU update processing but will also generate a GPU interrupt request.

Information relevent to other elements may be examined by stepping up or down from an element's PWC, using the resultant count to load SELWC, and then running with Select-halt-enable ON (CTL[09]) and Select Enable OFF (CTL[121]).

The above uses of the "interaction-aid" options allow correlating a user's logical data structure (object number, element count) with the physical description (memory address), with the display file instances (picture element count, picture word count), and with target data (at refresh buffer locations given by RBU counts).

4-3.   USE OF A DISPLAY OBJECT

Figure 4-1 illustrates a display of one object, a triangle drawn in the center of the display screen. The following paragraph describes a sample Object List, a sample directory, and a sample Stack Space which could be used to generate the refresh list to display the object.
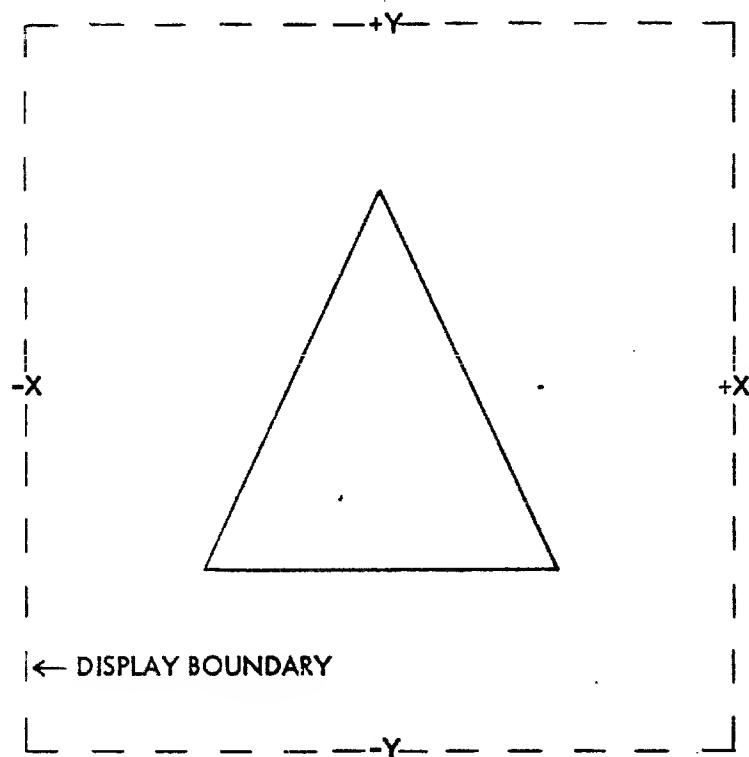
Figure 4-1. Example of One displayed Object

4-2

Figure 4-2 illustrates the sample Object List (TRI OBJ), sample Directory starting at location 0064 and a sample Stack area of 200 words which may be used to display the triangle. The stack area is not used in this example. The contents (0001) of Directory address 0064 specify that the Directory contains only one object.

SAMPLE DIRECTORY			TRI OBJ		
DIR 0064:	0001	One object	0078:	0001	No locals or links (next word is 1st instruction)
TRI ADDR:	0078	Triangle address		C068	LINES! LFIT! DFWD! BMJL! CAX! CAY! CCZ
				C000	-HALF X start point
				C000	-HALF Y start point
				4000	+HALF X end point
				C000	-HALF Y end point, (drawn)
OVERFLOW STACK SPACE				0000	∅ X end point
STACK 00C8:		(stack base)		4000	+HALF Y end point, (drawn)
	200			C000	-HALF X end point
				C001	-HALF! TERMN Terminal Y endpoint (drawn)
STKMAX 0190:		(stack limit)		1000	RETU

Figure 4-2. Sample Directory, Stack Space, and TRI Object

The computer code used to display the triangle is as follows:

Programmed Outputs:		Value Sent to GPU:
GPU [STB.]	← #STACK	00C8
GPU [SLM.]	← #STKMAX	0190
GPU [DIR.]	← #DRCTRY	0064
GPU [PBO.]	← #1	0001

Start Display: GPU [CMD.] ← NPIC! GO! DCUIN! STA1! DBM   C050

The Display will now access TRI OBJ, process the LINES instruction, and terminate the update pass when the unconditional return RETU is encountered at the end of the object list. The command codes specify: New Picture, Start Processing, Initialize Display Control Unit, Generate Refresh List at Station (RBU/DCU) #1, and Process Output in the Double Buffered Mode.

4-4. USE OF SUB-OBJECTS

The triangle and box drawn in Figure 4-3 illustrate an example of two sub-object calls from a "main picture" object. Figure 4-4 illustrates the Directory, PIC OBJ and the BOX Sub-object display lists. The TRI OBJ list remains unchanged; however, it also is called as a sub-object by PIC OBJ.
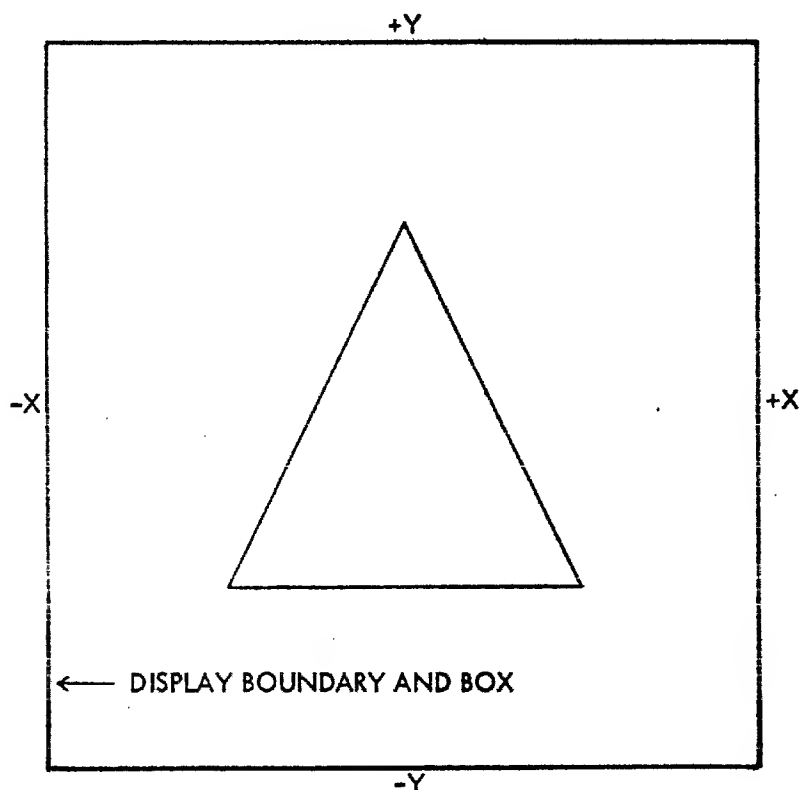
Figure 4-3. Example of Sub-Objects

The code used to display the two sample objects in Figure 4-3 is identical with that illustrated on page 4-3 with one exception: the picture base object register (PBO) is loaded with 0002 (Index of PIC in DIR). PIC is then accessed first, and calls are made from it to both BOX and TRI.

SAMPLE DIRECTORY			TRI OBJ	
DIR 0064:	0003	Three Objects	0078:	0001
TRI ADDR:	0078	Object TRI		
PIC ADDR:	008C	Object PIC		SEE
BOX ADDR:	00B0	Object BOX		FIGURE 4-1

PIC OBJ			BOX OBJ		
008C:	0003	SKIP OVER 2 LINKS	00B0:	0001	(NO LOCALS OR LINKS)
	0001	;INDEX OF TRI IN DIR		C0A8	LINES! LFIT! DFWD! BMHV! CAX!CAY!CCZ
	0003	;INDEX OF BOX IN DIR		8000	-FULL X start point
	6002	CALL 2 ;CALL BOX (2ND LINK)		8000	-FULL Y start point
	6001	CALL 1 ;CALL TRI (1ST LINK)		7FFE	+FULL X endpoint (drawn)
	1000	RETU		7FFE	+FULL Y endpoint (drawn)
				8000	-FULL X endpoint (drawn)
				8001	-FULL ! TERMN. Terminal Y endpoint (drawn)
				1000	RETU

Figure 4-4. Example of Sub-Object CALLS

4-5. USE OF NESTING TO APPLY TRANSFORMATIONS

Figure 4-5 illustrates a display of the TRI and BOX objects described above in which scaling and displacement transforms are applied to the objects by use of the Nest Immediate instruction (NESTI). In this example, no changes are required for either the Directory or the object lists for TRI and BOX. The only object list which requires changes is PIC OBJ (refer to Figure 4-6). The BOX is first scaled, displaced in a southwest direction, and drawn. The BOX is again scaled, displaced in the southeast direction, and redrawn. Finally, TRI is displaced in a +Y direction and drawn.

NOTE ON PROGRAMMING PRACTICE

Nesting of sub-objects is a tool for preserving the structure of data. Modules may be altered (i.e., scaled) or replaced without explicitly operating on affected element parameters (i.e., line endpoint coordinates). Dynamic or data-dependent use of nesting parameters permits representation of parametric displays. The sub-objects could be moving vehicles across a terrain, cams and links in machinery, or selected figures being interactively adjusted and placed in a drawing with an on-line drawing package.

The use of nested transforms to compose elementary static figures in these examples is for illustrative purposes only and is not recommended as effective programming practice for definition of unstructured elements.
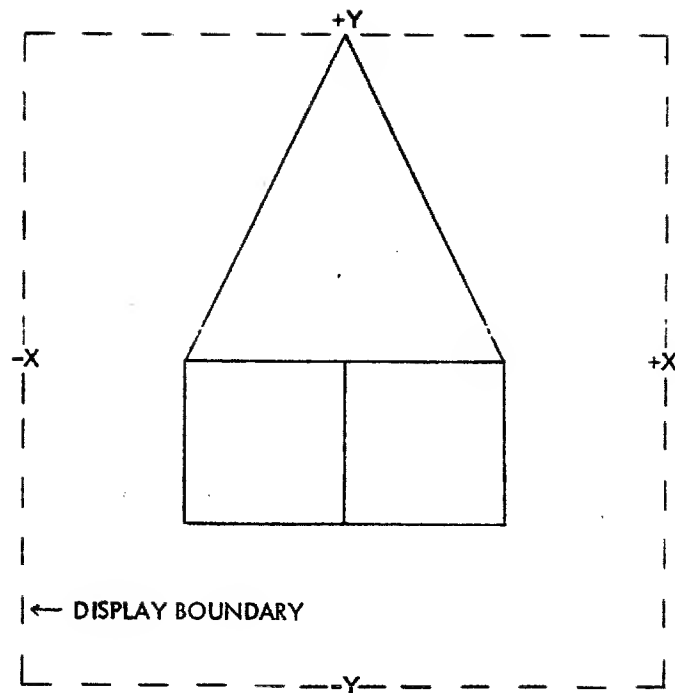
Figure 4-5. Example of Scaling and Displacement Through Nesting

PIC OBJ			
008C:	0003	;SKIP OVER 2 LINKS	
	0001	;INDEX OF TRI IN DIR	
	0003	;INDEX OF BOX IN DIR	
	5800	NESTI NOSXY	OBJECT SCALE, X AND Y DISPLACEMENTS
	2000	FOURTH; SCALE	
	E000	−FOURTH; DX	1ST TRANSFORMATION FOR BOX
	E000	−FOURTH; DY	
	6002	;CALL BOX (2ND LINK)	
	5800	NESTI NOSXY	OBJECT SCALE, X AND Y DISPLACEMENTS
	2000	FOURTH ;SCALE	
	2000	+FOURTH ;DX	2ND TRANSFORMATION FOR BOX
	E000	−FOURTH ;DY	
	6002	;CALL BOX (2ND LINK)	
	5802	NESTI NODXY	OBJECT X AND Y DISPLACEMENTS
	0000	Ø ;DX	TRANSFORMATION FOR TRI
	4000	+HALF ;DY	
	6001	;CALL TRI (1ST LINK)	
	1000	RETU	

Figure 4-6. PIC OBJ Example of Nesting and Transformation

4-6

4-6. USE OF DATA SCALE

Assume that it is desired to rotate the triangle displayed in Figure 4-5 and to place the inverted image below the entire assemblage, thereby generating a display as shown in Figure 4-7. Note that the assemblage of transformed objects would extend out of the "page" in which the construct has been defined. This "page size" is called the "dynamic range" of the construct. To define constructs adequately with rotated and/or displaced sub-objects, the dynamic range must be sufficient to encompass all elements at their largest coordinate values. To accomplish this, all elements of a construct may be scaled down by a Data Scale value in the GPU [DS].

NOTE: System resolution of 16 bits (14 bits if clipped)applies to the dynamic range used by any construct. Thus, setting Data Scale too small will waste resolution by needlessly reducing the accuracy available for element definition.
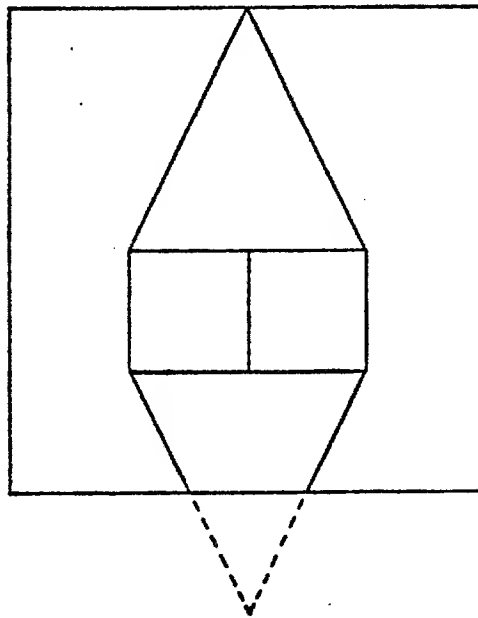
Figure 4-7. Example of a Construct Exceeding Dynamic Range

To adjust the dynamic range for the objects displayed in Figure 4-7, Data Scale could be applied to the entire display output by setting it once (via a Programmed Output to GPU [DS] during start-up). However, since Data Scale is set only for PIC OBJ (2/3 DS to correct this example), it would be more appropriate to include (and limit) its setting within PIC as shown in PIC1 (refer to Figure 4-8). The GPU will automatically compensate the viewing scale, and the display will remain as illustrated in Figure 4-7.

As in the previous example, no alteration is required to either the Directory or to object lists of TRI and BOX. Only the Picture Base Object is changed for applying Data Scale, Rotation and Displacement to TRI.

4-7.   USE OF VIEWING PARAMETERS

The displayed view of a construct may be manipulated without altering its definition by use of the following two facilities:

a)   The Windowing parameters may be used to extract a rectangular portion of the construct as the portion for display processing, and

b)   The Picture parameters may be used to place and limit the display on the screen.

	PIC1 OBJ		
008C:	0003	;SKIP OVER 2 LINKS	
	0001	;INDEX OF TRI IN DIR	
	0003	;INDEX OF BOX IN DIR	
**	4801	;LOAD IMMEDIATE 1 VALUE	
**	0020	;GPU [DS] DESTINATION ADDRESS	
**	5555	;2/3 DATA SCALE	
	5800	NESTI NOSXY	OBJECT SCALE, X AND Y DISPLACEMENTS
	2000	FOURTH ;SCALE	
	E000	-FOURTH ;DX	1ST TRANSFORMATION FOR BOX
	E000	-FOURTH ;DY	
	6002	;CALL BOX (2ND LINK)	
	5800	NESTI NOSXY	OBJECT SCALE, X AND Y DISPLACEMENTS
	2000	FOURTH ;SCALE	
	2000	+FOURTH ;DX	2ND TRANSFORMATION FOR BOX
	E000	-FOURTH ;DY	
	6002	;CALL BOX (2ND LINK)	
	5802	NESTI NODXY	OBJECT X AND Y DISPLACEMENTS
	0000	Ø ;DX	1ST TRANSFORMATION FOR TRI
	4000	+HALF ;DY	
	6001	;CALL TRI (1ST LINK)	
**	5808	NESTI NODY	OBJECT Y DISPLACEMENT
**	8000	-FULL ;DY	2ND TRANSFORMATION FOR TRI
**	580C	NESTI NRZ	Z ROTATION VALUE
**	8000	180° Z ROTATION	
**	6001	;CALL TRI (1ST LINK)	
**	4801	;LOAD IMMEDIATE 1 VALUE	
**	0020	;GPU [DS] DESTINATION ADDRESS	
**	7FFF	FULL ;DATA SCALE	RESTORE DATA SCALE
	1000	RETU	

**CHANGES TO PIC OBJ FOR 2/3 DATA SCALE AND Z ROTATION

Figure 4-8. PBO For Example in Paragraph 4-6.

Figure 4-9. (Deleted)

4-9

4-8. WINDOWING

As an example of the use of windowing, refer to the "over-range" construction in
Figure 4-7. In reducing Data Scale to 2/3 to permit the construct to fit the dynamic
range, the viewing scale was adjusted to retain the same display scale. The entire
construct can be displayed (as shown in Figure 4-10) by enabling the clipping feature
and by setting the Window Scale to full-scale.
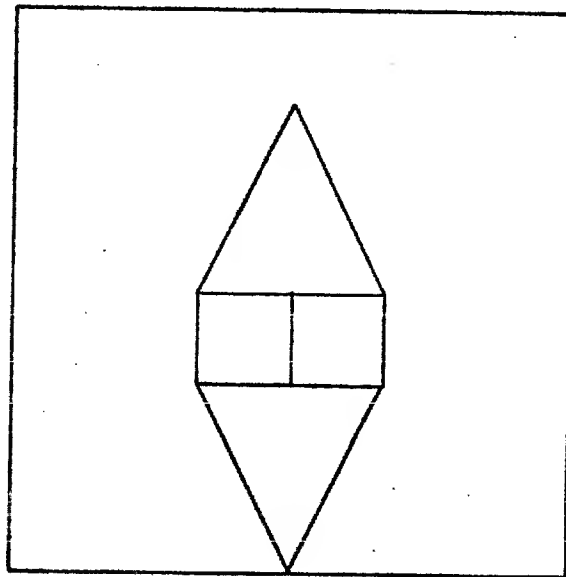
Figure 4-10. Example of "Zooming Out" and Clipping

4-9. Zooming

The visual effect of construct magnification or reduction can be presented by "zooming
in" or "zooming out," respectively. This is accomplished by reducing the Window
Scales (WSX, WSY Registers) for "zooming in" or by increasing the Window Scales
to "zoom out." Table 4-1 lists a routine that "zooms in" on the construct to produce
the example shown in Figure 4-10.

Table 4-1. Routine for Zooming In

Description	Register	Value	Machine Code (Hex)
Establish Stack	GPU [STB.]	#STACK	00C8
	GPU [SLM.]	#STKMAX	0190
Establish Display:	GPU [DIR.]	#DIR	0064
	GPU [PBO.]	#2	0002
Establish View	GPU [WSX.]	#2/3	5555
	GPU [WSY.]	#2/3	5555
Start Display:	GPU [CMD.]	#NEW PIC! GO! STA1!	C054
	GPU	DBUF!CLPVEC	

4-10.   Panning

The extracted portion of a construct may be selected from anywhere in the construct's dynamic range by moving the Window Center Point (WCX, WCY). This has a viewing effect of moving the display in a direction on the display screen opposite to the actual movement of the Window Center Point. The following Programmed Outputs will display the view shown in Figure 4-11.

GPU [WCX.]   5000; 5/8 FULL SCALE

GPU [WCY.]   F000; -1/8 FULL SCALE

WINDOW AND DISPLAY BOUNDARY →
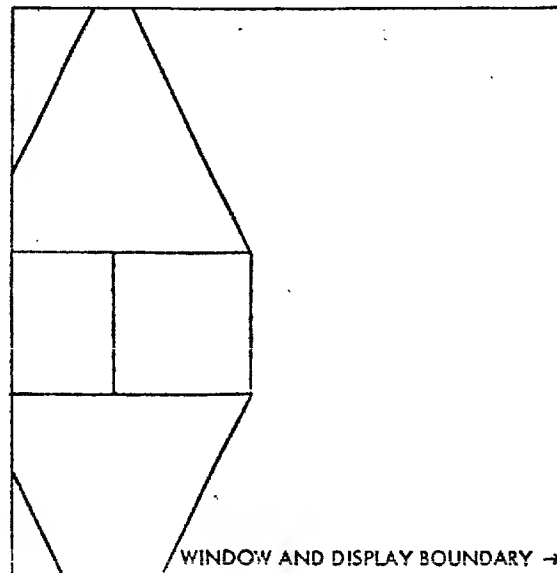
Figure 4-11. Example of "Panning"

4-11.   PICTURE TRANSFORMATIONS

At times it may be desired to use the display screen for more than one single display.
When areas of the screen are to be allocated to different presentations, the "Picture
Transformation" parameters are used to scale and place windowed views on the screen.
The different displays may be different views of the same object, menus, messages,
etc.   To allocate the display in Figure 4-11 to a "viewport located at the upper right
corner of the screen, the following picture parameter values would be loaded to dis-
play the screen as shown in Figure 4-12.

GPU [PS.]   4000; 1/2 FULL SCALE

GPU [PDX.]   4000; X CENTER + 1/2

GPU [PDY.]   4000; Y CENTER + 1/2
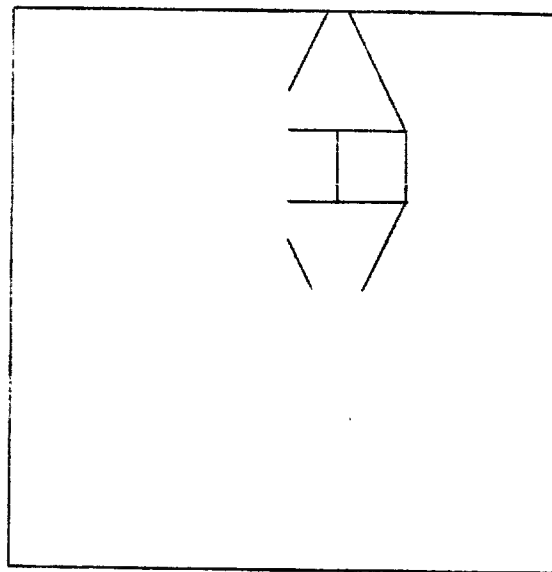
Figure 4-12. Example of Picture Transformation of Windowed View

4-12. SUMMARY OF VIEWING TRANSFORMATIONS

The diagrams in Figure 4-13 illustrate a summary of viewing transformations.

where:				
Window Center $\overline{WC}$	= (WCX,WCY)	←	4000,2000 ;+HALF X,+FOURTH Y
Window Scale $\overline{WS}$	= (WSX,WSY)	←	2000,4000 ;FOURTH X,HALF Y
(2D) Picture Displacement $\overline{PD}$	= (PDX,PDY)	←	E000,2000 ;-FOURTH X,+FOURTH Y
Picture Scale PS	=	PS	←	3000 ;3/8 FULL SCALE
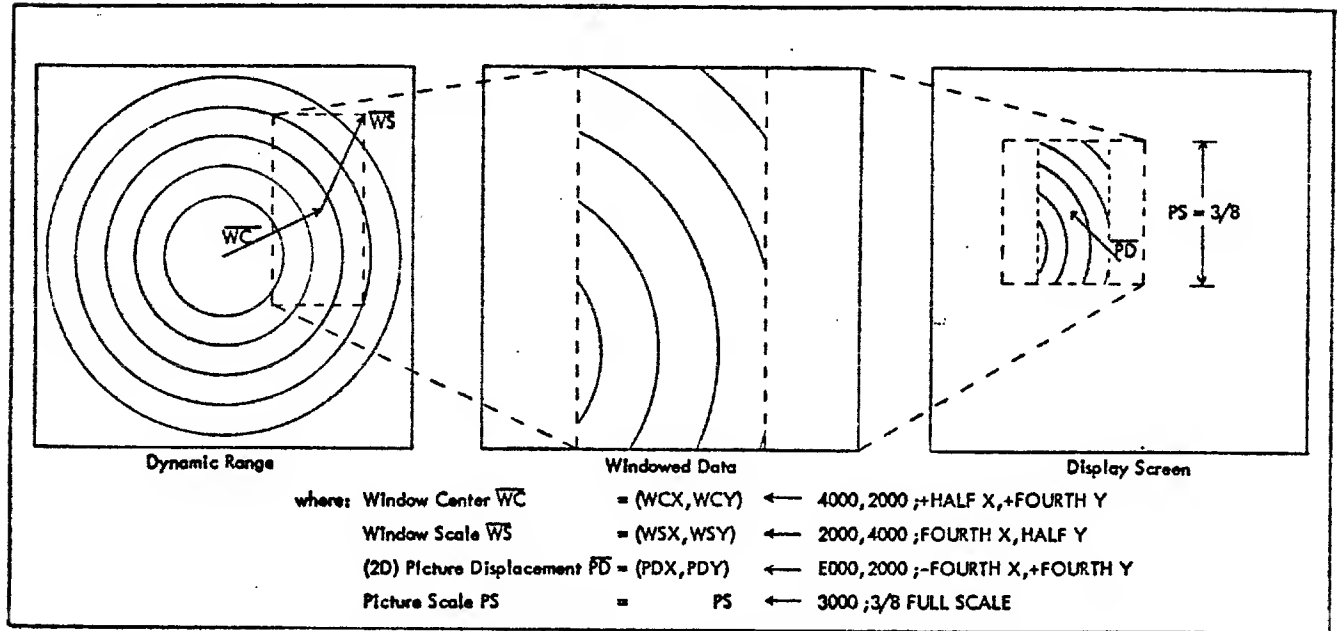
Figure 4-13. Summary of Viewing Transformations

4-13. THREE-DIMENSIONAL VIEWING PARAMETERS

The following sequence of illustrations presents the effect of viewing parameter alterations upon the display of 3-dimensional data.

The data define a full size cube containing three objects: a shallow box towards the front left, a pyramid in the center, and a prism in the right rear of the cube.

When update processing is started by sending the GO and NEW PICTURE bits to the CMD register, all transformations are reset and viewing is initialized as follows:

> Data window at maximum (no zoom magnification): WSX - WSY = Full Scale.
> All data are included in window (no front/rear cuts): WNX = WSZ = Full Scale.
> Window centered in data volume (no panning): WCX = WCY = Zero.
> Viewport centered on screen: PDX = PDY = Zero.
> Viewpoint at Z = infinity (no perspective depth cue): PDZ = Zero.
>
> Z viewpoint = [WNZ + max (WSX, WSY)]/PDZ.

With these values the data will be displayed in an unrotated orthogonal view as illustrated in Figure 4-14.
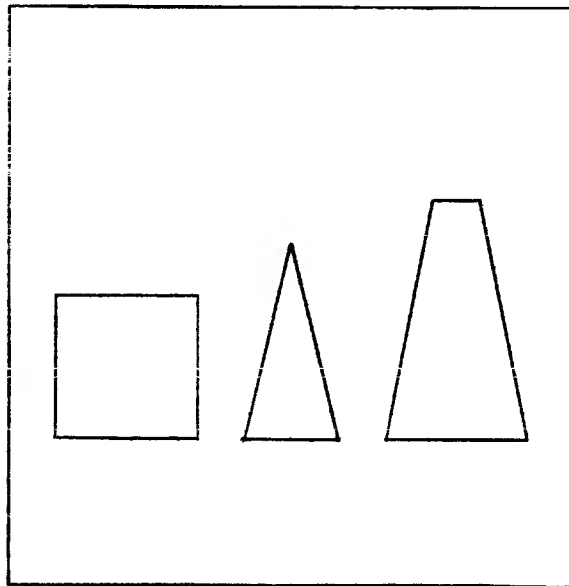
Figure 4-14. Initialized View

$\overline{\text{PD}}$:	0	0	0
$\overline{\text{WC}}$:	0	0	
$\overline{\text{WNZ}}$:			FS
$\overline{\text{WS}}$:	FS	FS	FS

The first change imposed is to increase PDZ to Full Scale, bringing the viewpoint in from Z= ∞ to Z= twice Full Scale (to Full Scale away from the front of the data). This causes data points behind the front plane (at WNZ = +Full Scale) to be foreshortened in Z and Y by their Z depth, creating the perspective projection shown in Figure 4-15.
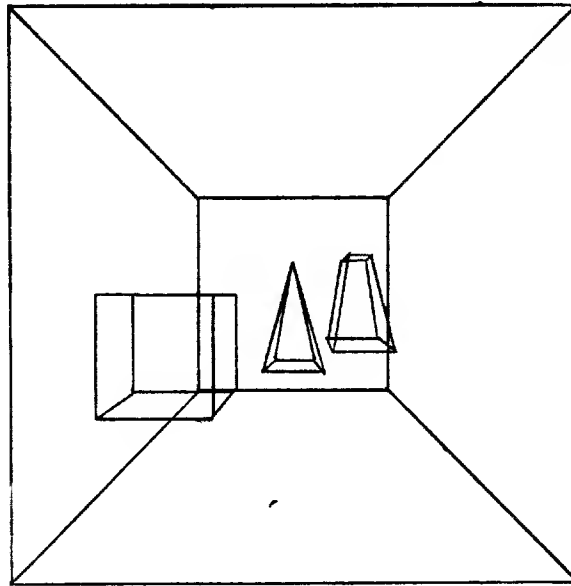
Figure 4-15. Perspective Cue

PD:	0	0	FS
WC:	0	0	
WNZ:			FS
WS:	FS	FS	FS

In the following view the presented data is magnified by shrinking the window volume to half its previous X-Y extent. This zooms in on the center portion of the displayed data as shown in Figure 4-16. Note that part of the data did not fit and was clipped.

In Figure 4-17 the position of the window is moved left bringing all of the data back into view and down to better resolve the top of the box. This is done with the window center registers as shown in Figure 4-17.
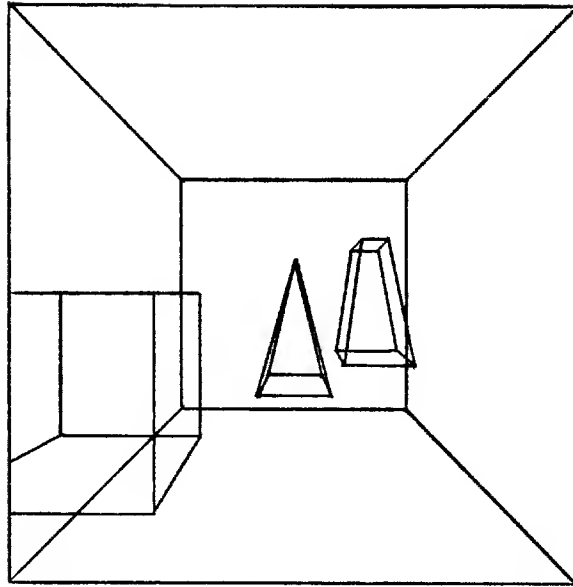
Figure 4-16.   Example of Zooming

$\overline{\text{PD}}$:	0	0	FS
$\overline{\text{WC}}$:	0	0	
$\overline{\text{WNZ}}$:			FS
$\overline{\text{WS}}$:	FS/2	FS/2	FS
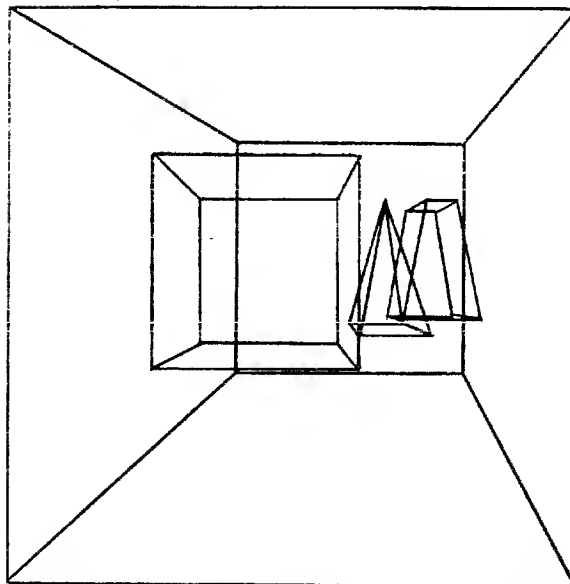
Figure 4-17.   Example of Panning

$\overline{\text{PD}}$:	0	0	FS
$\overline{\text{WC}}$:	-FS/2	-FS/3	
$\overline{\text{WNZ}}$:			FS
$\overline{\text{WS}}$:	FS/2	FS/2	FS

The following pair of changes move the front and rear of the window volume and illustrate the "decluttering" of a display. First, the front-cut plane is moved to a position between the box and pyramid, thereby eliminating the box. This is done by reducing WNZ to 1/3 Full Scale (see Figure 4-18).
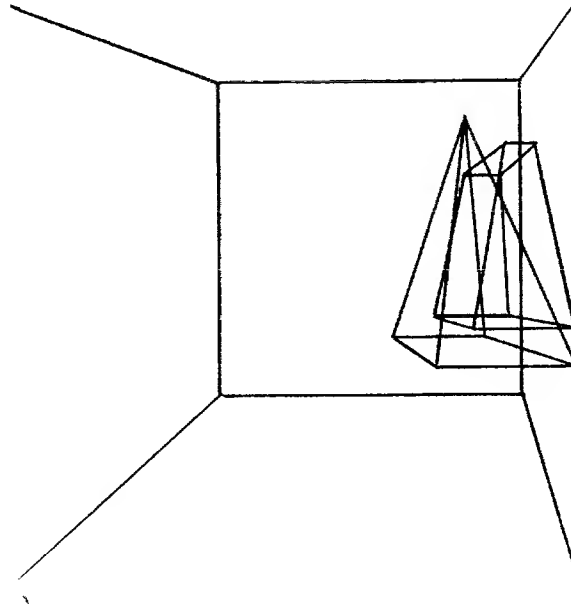
Figure 4-18.   Example of "Front Cut"

$\overline{PD}$:   0   0   FS
$\overline{WC}$: -FS/2   -FS/2
$\overline{WNZ}$:   FS/3
$\overline{WS}$:   FS/2   FS/2   FS

In the last example, the rear cutoff plane is brought forward by reducing the window depth to half of that possible with the current WNZ setting. The window will no longer include the prism (nor the rear of the full-scale cube), resulting in a display as illustrated in Figure 4-19.
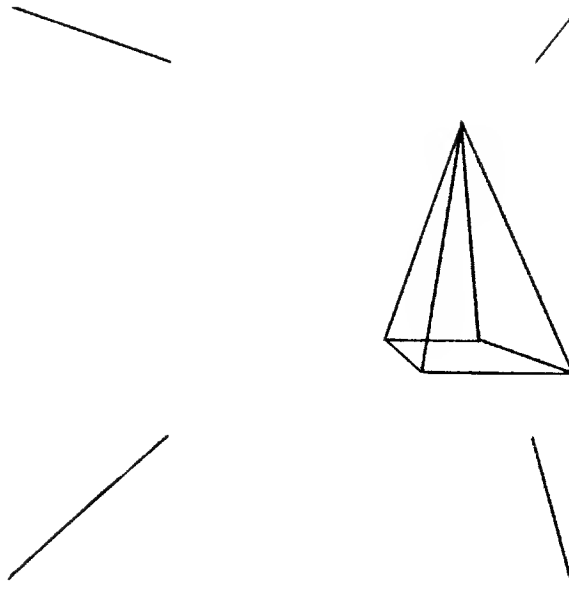
Figure 4-19.   Example of Both Front and Rear Cuts

$\overline{PD}$:   0   0   FS
$\overline{WC}$: -FS/2 -FS/3
$\overline{WNZ}$:   FS/3
$\overline{WS}$:   FS/2   FS/2 FS/2

4-14. ARGUMENT ADDRESSING EXAMPLE

A display of a typical PERT chart (see Figure 4-20), and an example of the coding which may be used to generate the display refresh list, is used here to determine some of the advantages and flexibility of the addressing techniques used in the 3400 system. The chart could be used to delineate a given sequence of tasks, and the time frame allotted to each, for scheduling the design, manufacture, and test of a typical new product. It then becomes a network in which each node represents the completion of a previously scheduled task and the start of the task identified by that node.
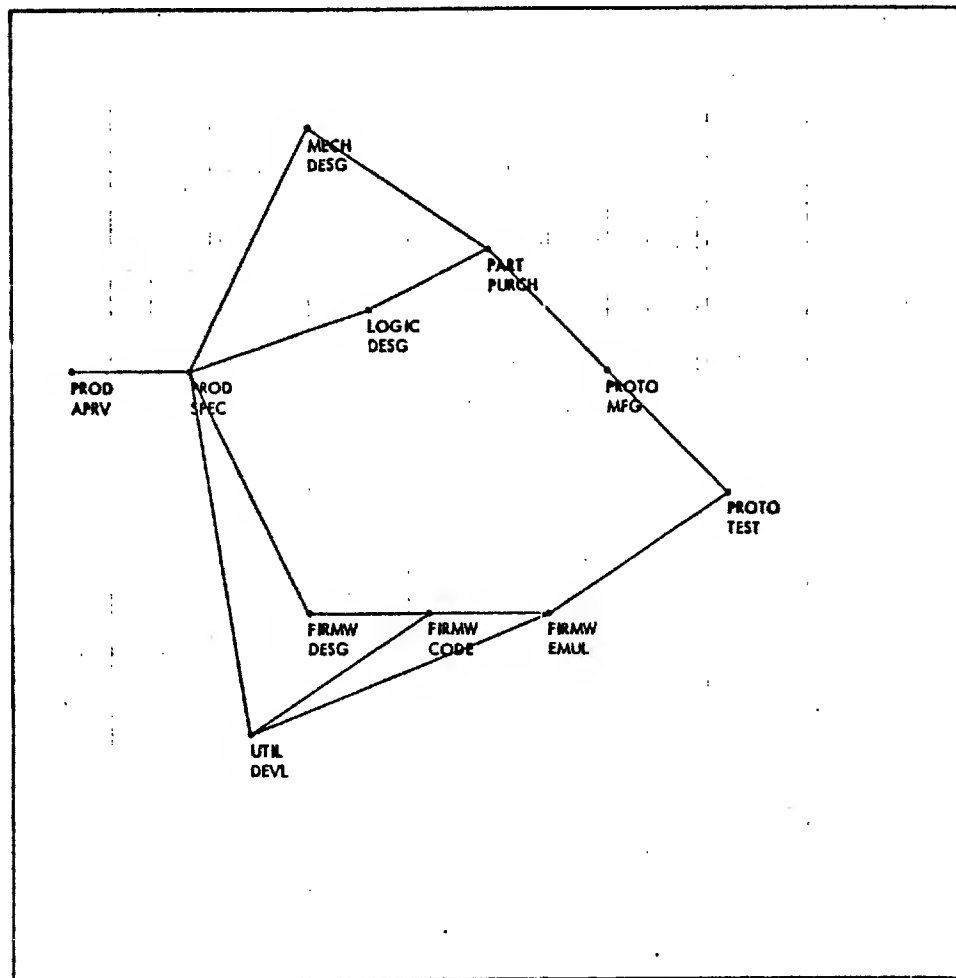
Figure 4-20. Display Example Demonstrating the Use of Addressing

In this example, three user data tables (refer to Table 4-1a, 4-1b and 4-1c) and two sub-objects (refer to Table 4-3) are used to define the display as follows:

Table 4-2. User Data Tables For Display Example

(a)	(b)	(c)

NODES: 100:

Ø	Ø	Ø	B	
9	Ø	Ø	Ø	1
2	Ø	Ø	Ø	2
B	Ø	Ø	Ø	3
2	Ø	Ø	Ø	4
D	Ø	Ø	Ø	5
6	Ø	Ø	Ø	6
E	Ø	Ø	Ø	7
3	Ø	Ø	Ø	8
D	Ø	Ø	Ø	9
E	Ø	Ø	Ø	A
C	Ø	Ø	Ø	B
C	Ø	Ø	Ø	C
Ø	Ø	Ø	Ø	D
4	Ø	Ø	Ø	E
F	Ø	Ø	Ø	F
E	Ø	Ø	Ø	10
1	Ø	Ø	Ø	11
E	Ø	Ø	Ø	12
2	Ø	Ø	Ø	13
2	Ø	Ø	Ø	14
4	Ø	Ø	Ø	15
Ø	Ø	Ø	Ø	16

NAMES: 200:

•| PROD | APRV	1
•| PROD | SPEC	2
•| MECH | DESG	3
•| LOGIC | DESG	4
•| FIRMW | DESG	5
•| UTIL | DEVL	6
•| PART | PURCH	7
•| FIRMW | CODE	8
•| FIRMW | EMUL	9
•| PROTO | MFG	A
•| PROTO | TEST	B

Notation: | = Carriage Return
           • = Node Character

BRNCHS: 400:

D
1
3
3
5
3
7
3
9
3
B
5
D
7
D
9
F
B
F
D
13
F
11
B
11
13
15
11
15

Table NODES   The first entry in this table is the number of nodes in the netword (in this case 11). This entry is followed by 11 coordinate-pairs (X and Y) which specify the current location of each node in the display. These locations serve two purposes:

a) They are used to locate the position of the node and associated text on the display, and

b) They are used as vector start and endpoints for the interconnecting lines.

Table NAMES   This user table contains a list of the text labels which are to be drawn at each node.

Table BRNCHS   The first entry in this table is the number of inter-node connecting lines to be displayed (in this case 14). This entry is then followed by 14 "number-pairs" which identify each pair of nodes to connect.

4-20

Object JOBS   This sub-object is used to display the 11 nodes and labels.

Object NET   This sub-object is used to display the 14 inter-node connecting lines.

4-15. OBJECT JOBS

The purpose of this object is to display each of the labeled nodes at its current position. Each node and label is displayed as a text string. The test strings are taken from the external table NAMES. The length of the text string is taken as a local variable NAML, to be set as an argument by the caller of JOBS. The position from the text strings is taken from the external table NODES.

The object can thus be written (refer to Table 4-3) as <u>one</u> display generating instruction: A TEXT instruction with its list of text data referenced externally, terminated by a count referenced as a local, and its page positioned by externally referenced coordinates. The data referenced externally is taken from tables indexed by GPU general purpose registers GP2 and GP3. The number of nodes displayed is counted down in GP1 and tested before each TEXT instruction to exit when no more remain.

One arithmetic instruction (GADD) is used to step the NAMES index GP2 by the name-length NAME, and one (GSUB) is used to count down the number of nodes (GP1) after each is processed. The coordinate index $GP\mu$ is stepped after each use in the "register decrement" mode thus sequencing through the NODES table.

The exhausted (or initially empty) NODES table (index GP1) is tested at the beginning of the loop by the RETZ instruction which will exit when GP1 = 0.

Prior to entering the loop the two indices and the loop-counter are initialized by three LOAD instructions.

4-16. OBJECT NET

The purpose of object NET is to display a line between each pair of nodes listed in table BRNCHS. This object also contains only one display-generating instruction: a LINES instruction whose coordinate list is referenced as a temporary storage in the local stack. The list length is referenced as an immediate count = 4 words.

Table 4-3. Display Example Source Code

```
DIRECTORY (JOBS, NODES, NAMES, NET, BRNCHS, PIC)

JOBS:   OBJECT   EXTERNAL(NODES, NAMES), LOCAL(NAML)
        LOAD     1,(REG GP2.),(IMD Ø)              ; index to NODES table
        LOAD     1,(REG GP3.),(IMD Ø)              ; index to NAMES table
        LOAD     1,(REG GP1.),(EXV $NODES)         ; count of remaining nodes
JOB1:   RETZ                                       ; quit if no more
        TEXT     LFRC, DFBY, PGXY                  ; display-positioned NODE-NAME
        EXI      $NODES,(RG1 GP2.)                 ; X-position (use & step index)
        EXI      $NODES,(RG1 GP2.)                 ; Y-position (use & step index)
        LOC      $NAML                             ; chars per name
        EXI      $NAMES,(REG GP3.)                 ; text of name
        GADD     (REG GP3.),(REG GP3.),(LOC $NAML) ; step NAME index
        GSUB     (REG GP1.),(REG GP1.),(IMD 1)     ; count down
        BRKL     JOB1 - JOB2                       ; loop back

JOB2:        .

NET:    OBJECT   EXTERNAL(BRNCHS, NODES)
        LOAD     1,(REG GP1),(EXV $BRNCHS)         ; load count of # of branches
        GADD     (REG GP1),(REG GP1),(REG GP1)     ; double to index node-pairs per branch
NET1:   RETZ                                       ; quit if no more
        LOAD     2,(REG GP2.)                      ; load reg 2 and 3 with
        EXI      $BRNCHS,(RGD GP1.)                ;   source node # and
        EXI      $BRNCHS,(RGD GP1.)                ;   destination node # (stepping index to next branch)
        PUSH     4                                 ; fetch endpoint coordinates to stack
        EXI      $NODES,(REG GP2.)                 ;   source node X
        EXI      $NODES,(RG1 GP2.)                 ;   step and get source node Y
        EXI      $NODES,(REG GP3.)                 ;   destination node X
        EXI      $NODES,(RG1 GP3.)                 ;   step and get destination node Y
        LINES    LFRC, BMDJ, CAX, CAY              ; draw arc: disjoint line abs X & Y, refr'd list w/count
        IMD      2                                 ;   count: two points
        ARG      1                                 ;   list refr: first temp cell in stack
        POP                                        ; discard stack
        BRKL     NET1 - NET2                       ; loop back
NET2:

PIC:    OBJECT   EXTERNAL(JOBS, NET)
        LOAD     1,                                ; set chars-per-name arg for JOBS
        EXI      $JOBS,(IMD $NAML)                 ;  NAML local to JOBS set to
        IMD      B                                 ;  eleven (hex B)
        CALL     $JOBS                             ; display positioned & named nodes
        CALL     $NET                              ; display all interconnecting branches
        RETU                                       ; end of display
```

The endpoint coordinates for the line are moved into the stack by the PUSH instruction. The values are referenced in external table NODES as indexed by GP2 for the line startpoint and GP3 for the line endpoint. After using each index to obtain an X coordinate, it is incremented by RGI mode prior to use in fetching the following Y coordinate.

The indices for the start and endpoint nodes are put into GP2 and GP3 by a LOAD instruction. It obtains them from table BRNCHS an indexed by GP1. After each node-index of a pair is fetched, GP1 is decremented to reference the next (in decreasing sequence) by "register decremented" mode.

The exhausted (or initially empty) BRNCHS table (index GP1) is tested at the beginning of the loop by the RETZ instruction.

Prior to entering the loop the BRNCHS table index is initialized to index the end of the table.

After each line is processed the local stack (created by the PUSH 4 instruction) is discarded by the POP instruction.

4-17.   OBJECT PIC

The two subpictures are called by object PIC which first sets up the name-length argument for sub-object JOBS.

This example illustrates use of some of the addressing modes to permit the following:

a)   Any node can be moved by just changing its coordinates in the NODES table. The node dot, its text label, and all attached branches will change appropriately.

b)   Nodes can be added and removed by adding or deleting the location from NODES and label from NAMES and adjusting the node count (first entry in NODES).

c)   Connecting links can be changed, added, or removed by just changing BRNCHS table and adjusting its count (first entry).

NOTE: These changes may be desired if the presentation is to be made available at an interactive terminal and none of them require any change of the coded display description. Thus addressing flexibility can be used to code more effective interactive display applications.

Table 4-4 (object Code of Addressing Example) and Table 4-5 (Macros Used in Addressing example) are included here to provide the reader with further information for reference.

Table 4-4. Object Code of Addressing Example

```
DIR: 500:  |        6 |                  JOBS: 600:  |        4 |                    NET: 7øø:  |        3 |
           | 6 ø ø | 1 JOBS                          |        2 | 1 NODES                          |        5 | 1 BRNCHS
           | 1 ø ø | 2 NODES                         |        3 | 2 NAMES                          |        2 | 2 NODES
           | 2 ø ø | 3 NAMES                                    3 NAML                  | 4 ø ø 1 |   LOAD 1
           | 7 ø ø | 4 NET                | 4 ø ø 1 |   LOAD 1                          | 3 ø ø F |     GP1 ← (BRNCHS)
           | 4 ø ø | 5 BRNCHS            | 3 ø 1 ø |     GP2 ← ø                        | C ø ø 1 |
           | 8 ø ø | 6 PIC              | ø ø ø ø |                                     | 8 ø ø ø |   GADD
                                        | 4 ø ø 1 |   LOAD 1                            | 3 ø ø F |     GP1 ← GP1 + GP1
                                        | 3 ø 1 1 |     GP3 ← ø                         | 3 ø ø F |
                                        | ø ø ø ø |                                     | 3 ø ø F |
                                        | 4 ø ø 1 |   LOAD 1                            | 1 8 ø ø |   RETZ
PIC: 800:  |        3 |                  | 3 ø ø F |     GP1 ← ($NODES)                 | 4 ø ø 2 |   LOAD 2
           |        1 | 1 JOBS          | C ø ø 1 |                                     | 3 ø 1 ø |
           |        4 | 2 NET           | 1 8 ø ø |   RETZ                              | 8 ø ø 1 |     GP2 ← BRNCH[GP1-]
           | 4 ø ø 1 |   LOAD 1        | E D C ø |   TEXT                               | 3 9 ø F |
           | 8 ø ø 1 |     EXI $JOBS[NAML] ← C    | 4 ø ø 3 |     $NAML               | 8 ø ø 1 |     GP3 ← BRNCH[GP1-]
           | ø ø ø 3 |                | 8 ø ø 2 |     $NAMES[GP3]                       | 3 9 ø F |
           | ø ø ø C |                | 3 ø 1 1 |                                       | 7 ø ø 4 |   PUSH 4
           | 6 ø ø 1 |   CALLU $JOBS   | 8 ø ø 1 |     $NODES[GP2+]                    | 8 ø ø 2 |     NODES[GP2]
           | 6 ø ø 2 |   CALLU $NET    | 3 1 1 ø |                                      | 3 ø 1 ø |
           | 1 ø ø ø |   RETU          | 8 ø ø 1 |     $NODES[GP2+]                    | 8 ø ø 2 |     NODES[+GP2]
                                       | 3 1 1 ø |                                      | 3 1 1 ø |
                                       | 8 ø ø ø |   GADD                              | 8 ø ø 2 |     NODES[GP3]
                                       | 4 ø ø 3 |     GP3 ← GP3 + NAML                | 3 ø 1 1 |
                                       | 3 ø 1 1 |                                      | 8 ø ø 2 |     NODES[+GP3]
                                       | 3 ø 1 1 |                                      | 3 1 1 1 |
                                       | 8 ø ø 1 |   GSUB                              | C C 2 8 |   LINES,LFRC,CAX,CAY
                                       | 3 ø ø F |     GP1 ← GP1-1                     |        4 |
                                       | ø ø ø 1 |                                     | 2 8 ø 1 |
                                       | 3 ø ø F |                                     | 7 ø ø ø |   POP
                                       | 3 F E E |   BRKL                              | 3 F E E |   BRKL
```

Table 4-5.   MACROS Used in Addressing Example

1:      Source line: DIR: DIRECTORY (a,b,c,d)

Defines: location   contents   symbols defined

DIR:

4	
a	@a = * – DIR
b	@b = * – DIR
c	@c = * – DIR
d	@d = * – DIR

2:      Source line: OBJ: OBJECT EXTERNAL (a,b,c), LOCAL (d,e)

Defines: location   contents   symbols defined

OBJ:

5	
@a	$a = * – OBJ
@b	$b = * – OBJ
@c	$c = * – OBJ
Ø	$d = * – OBJ
Ø	$e = * – OBJ

NOTE: * = "current location value"
        @a and $a are symbol names generated from an argument "a".

4-18. INTERACTION-AID FEATURES

Interaction-aids described here include the SELECT-INTERACTION-AID (SELECT), the HIT-INTERACTION-AID (HIT), and the EDIT-INTERACTION-AID (EDIT-AID). The SELECT and HIT aids are used for displayed item selection and picking, while the EDIT-AID is used for tentatively viewing the effect of element deletions, insertions, and adjustments when considering an update of the display data base. In the SELECT aid, the GPU receives a selected word count (SELWC) from the program which may be used to highlight a selected item, while with the HIT aid a picked word count (HITWC) is derived interactively by the viewer/operator using optional PICK or PEN devices. Either of the two word counts may be used as the edit word count (EDWC) in the EDIT-AID feature to identify the start point of a deletion of a display list word sequence or the insertion of a trial word sequence.

4-19. SELECT FEATURE

With this feature, the program selects items on the display that are to be identified. Program selection of an item may be controlled by either the word count (which may be incremented or decremented by stepping through each successive word (PWC) in the display list) or by other inputs or computations specifying an item as illustrated in Figure 4-21.
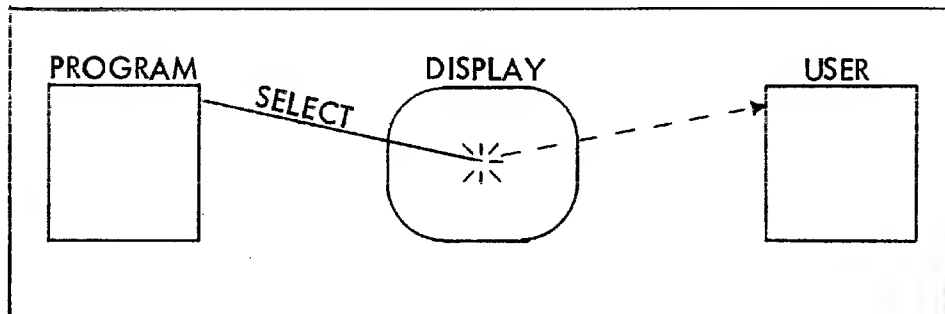
Figure 4-21. Item Selection

GPU hardware registers required to support the SELECT feature are as follows:

SELCT   -a select register, under control of the host computer, used to specify the mode of item highlighting (brighten, blink and brighten, etc.).

SELWC   -a 16-bit register, loaded by the host computer, which contains the word count of a currently selected item to be highlighted. If set to zero, the device (light pen or pick window) whose address is contained in HITDEV will automatically highlight a single display element (as specified by SELCT) when a hit is detected.

HITDEV   -a device select register, loaded by the host computer, which specifies the interactive device to be used for device auto-select.

4-20.   Application of the Select Feature

An example of choosing a displayed item using the keyboard and the SELECT option follows. Two program examples (STARTUP and KEY) are first described and then followed by a description of their use.

a) STARTUP: 1.   Set GPU SELWC to ÷1 (first word in the display list), SELCT to "blink bright," and HITDEV to 0.

2.   Set up keyboard interrupt to call the "KEY" routine described in (b).

3.   Reset software "done flag" in core (to be set later by KEY routine).

4.   Start GPU processing with SELECT ENABLE bit ON [CTL (12)].

5.   Wait for "done flag;" then disable keyboard and take SELWC as the operator specified picture word count of item chosen.

b) KEY: 1.   Read the typed character and test for the following three cases:

if ◄— is typed, decrement SELWC (to a minimum of +1).
if —► is typed, increment SELWC +1.
if "HOM" is typed, set "done flag."

c) Usage   To perform the selection run program "STARTUP." Press the "   " key on the keyboard; SELWC will be stepped to correspond with successive words in the display description being presented. As SELWC moves through words of visible items, the displayed image of such items will be highlighted by flashing brighter at a noticeable blink rate. If the desired item is passed, the " —► " key will decrement SELWC to return to the previous item. When the desired item is blinking, press the "HOM" key to designate choice of item.

NOTE: Depending on the construction of the display list, pressing the " —► " key will either move the highlighting to the right (as in test lists) or make random selection anywhere on the screen. Refer to paragraph 4-25 for suggested uses.

4-21. Implementation of the Select Feature

Figure 4-22 illustrates the display system information flow which is used to select a desired display item to be highlighted. Correlation between the selected item to be highlighted and the DCU highlighting of that selected item is accomplished by using word comparisons.

While the GPU is organizing the display list data from the host computer into the display refresh list to be stored in the RBU, it maintains a picture word count (PWC) as each data word is received from the CPU and an RBU word count (RBUWC) as each restructured word is entered into the RBU. The DCU also uses a word counter (WC1) which is incremented as it receives each data word from the RBU and, as a result, for any specific RBU word, the RBUWC and WC1 counts will be the same.

When the program specifies an item to be selected, its word count is placed into the GPU SELWC. When the PWC compares with the specified SELWC, the RBUWC from the GPU is loaded into a select register (SELR) in the DCU. The DCU then makes comparisons between SELR and each WC1 count as words are received from the RBU and, when a comparison occurs, it highlights the selected item.
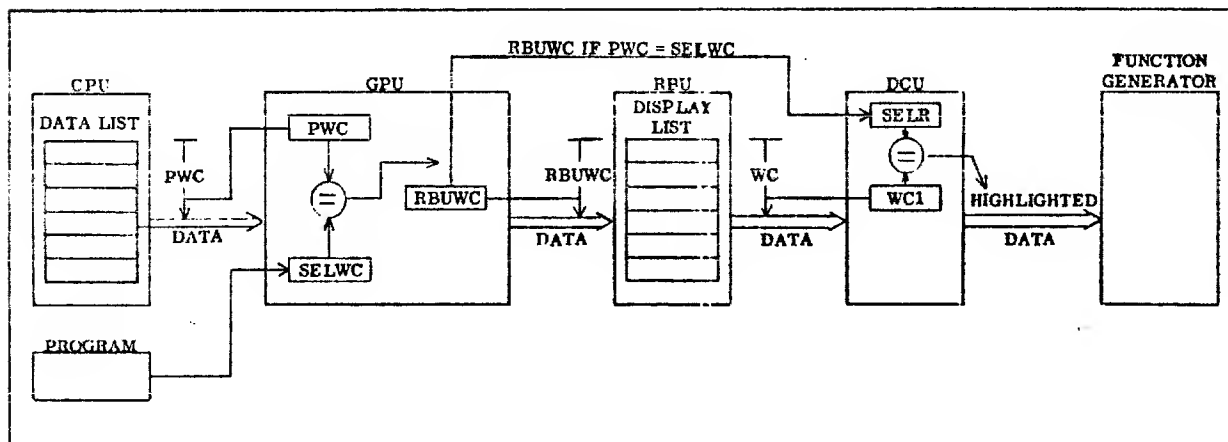
Figure 4-22. Implementation of the Select Feature

4-22. HIT FEATURE AND PEN OR PICK OPTION

With this feature, the user picks an item on the display screen and the hardware provides the program its picked word count (HITWC). Figure 4-23 illustrates the "picking" function. With the PICK option, a 2-D device "window" is generated and

4-28

and used to detect, and highlight if specified, elements that fall within the window boundaries. The window position, derived from interactive device X-Y positioning coordinates are specified by computer program. A combination of GPU/DCU operations is then used to notify the CPU of the picture word count (HITWC) of an item which is "picked." Refer to Paragraph 3-51 for descriptions of the GPU hardware registers required to support this option.
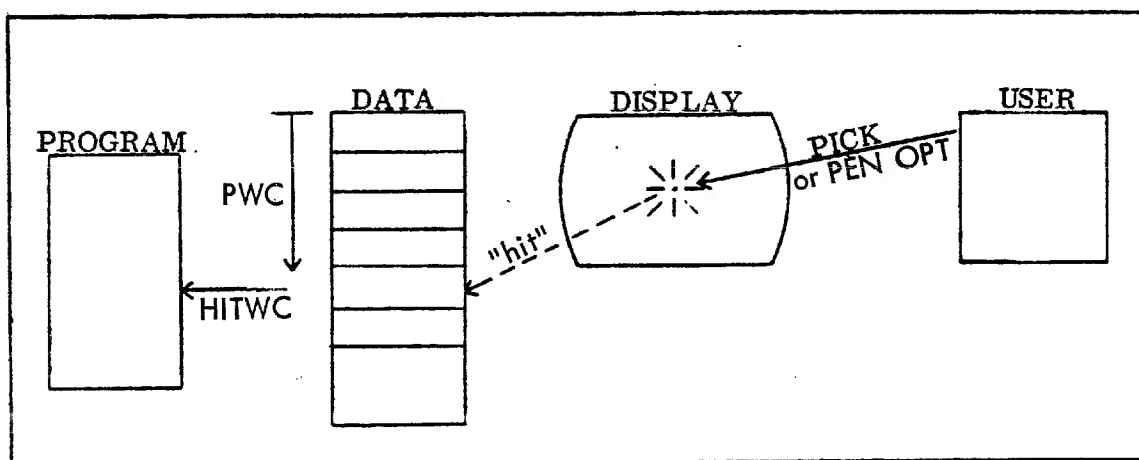
Figure 4-23. Item Picking

4-23. Application of HIT Feature With PICK Option and Data Tablet

An example of choosing a displayed item using the Data Tablet and the PICK option is presented below. The sample TABPIC program is first described and then followed by a description of its use.

a) TABPIC: 1. Set PICK window size (GPU register PIKS) to 1/50th screen size, HN to 1, SELCT to C000 and HITDEV to 0138.

2. Set HIT enable (CTL [13]) and start GPU processing.

3. Repeat this step until Tablet Stylus is pressed:
Read HITWC and send to SELWC.
Read Tablet X and send to GPU PIKX.
Read Tablet Y and send to GPU PIKY.

4. When Tablet Stylus is pressed, reset HIT enable and take HITWC and the PWC of the chosen item.

b) Usage   To perform the selection, run program "TABPIC." As the tablet
stylus is moved across the tablet surface, PIKX and PIKY are up-
dated to move the window correspondingly. When the displayed items
are within 1/4th inch of the stylus X-Y coordinates they will be high-
lighted by the SELECT facility. When the desired item is flashing,
press the stylus to activate the pen pressure switch and designate the
desired selection.

4-24.   Implementation of the HIT Feature

Figure 4-24 illustrates the implementation required to notify the program of the PWC
of an item "picked" by an interactive device. Similar to the SELECT option described
previously, the identity of the item is established through use of word count compari-
sons. Figure 4-24 illustrates the use of the light pen as the interactive device used
to signal a "hit." During the refresh cycle when the "hit" is detected, the RBU word
count (WC1) is placed by the DCU into the specified hit register (HIT) and also into a
specified select register (SEL). During a GPU update pass when a comparison be-
tween HIT and RBUWC occurs, the PWC is placed into the pick word count register
(HITWC) for transfer to the program where it is used to establish the PWC of the
"picked" item. Also, if specified by program, the SEL contents are used to highlight
the "picked" item.
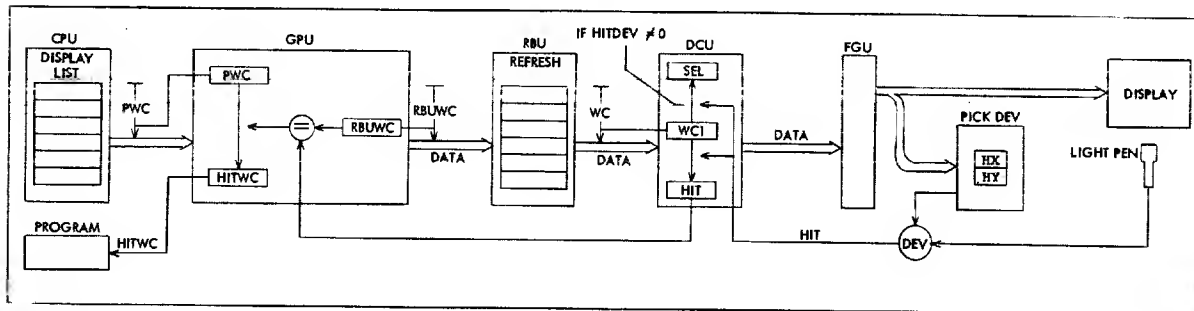
Figure 4-24. Implementation of Hit Feature

4-25.   SUGGESTED USES FOR SELECT AND HIT

Efficiency in the manner in which items on the display may be chosen depends, to
some extent, on the relationship between the order of data in the display list and the
sequence in which they are presented on the display screen. For example, if the
data order is displayed contiguously on the screen (such as text strings, data-plots,

etc.) the SELECT example in Paragraph 4-20 using the keyboard →and ← keys provides an efficient method of stepping quickly to the desired item. However, if the displayed data is random with respect to the display list, the HIT mode example in Paragraph 4-23 provides a means of choosing the desired item more rapidly.

Another possibility of rapid item-picking would be the use of the keyboard and the HIT feature where the keys → , ↑ , ↓ , ← , provide Δ X and ΔY values to update the GPU PIKX and PIKY registers and thus provide a corresponding direction of window movement toward the desired item.

4-26.   EDIT-INTERACTION-AID

This feature provides hardware for interactive (tentative) deletion, insertion, and adjustment of displayed elements while utilizing display visual feedback to observe their effects. This permits altering the display be substitution of arbitrary word sequences in the display list with trial word sequences which may be adjusted until visual satisfaction is achieved. The operator may then incorporate all changes into the display data base when an up-date is desired by calling his Data Base Update Application Program. Figure 4-25 illustrates the effects of substitution and updating the data base.

Figure 4-25A shows a display list data transfer to the GPU with no EDIT-AID used.

Figure 4-25B shows an example of a display word sequence being deleted and a trial word sequence from BUF sent to the GPU and the insertion of the new item. The data base words (alpha, beta, and gamma) are still sent to the GPU, however, the GPU skips the beta word and substitutes beta prime when received. During this time, the contents of BUF may be interactively adjusted until viewing satisfaction is received.

Figure 4-25C shows the data base update by permanently replacing the selected word sequence with the trial sequence from BUF. The update is accomplished by calling a user's Data Base Update Application Program when editing is done.

Paragraph 3-53 provides descriptions of the GPU registers required to support the EDIT-AID option.
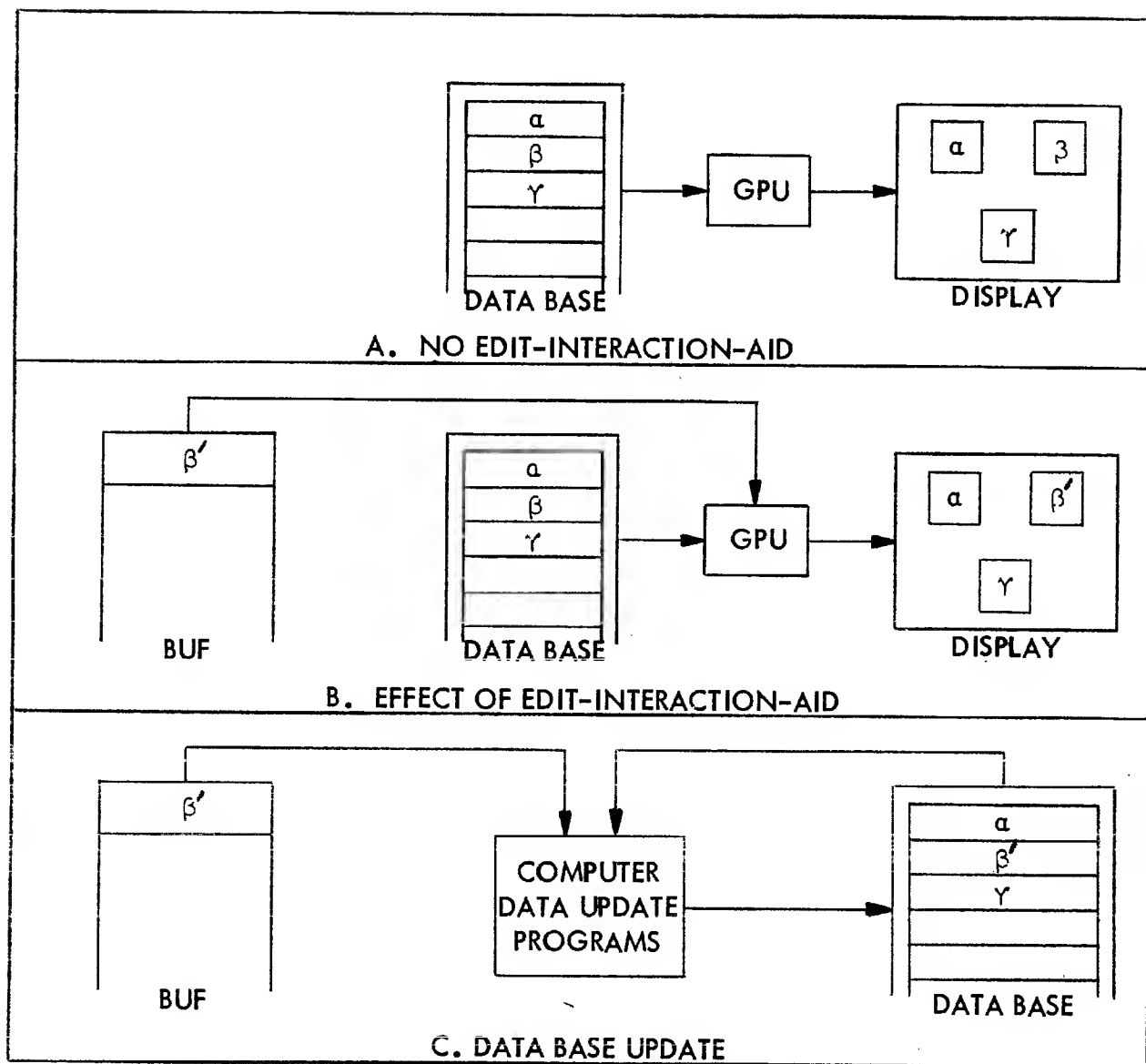
Figure 4-25. Use of Edit-Interaction-Aid

4-27.   DISPLAY WORD SUBSTITUTION

Deletion and insertion of items on the display through use of this option are described in the following paragraphs.

4-28.   Deletion

An example of a program for deletion of items ("DELETE"), permitting observation of viewing effects, is listed in Table 4-6.

Table 4-6.   Sequence for Sample DELETE Program

Step	Instruction
1	Load the GPU EIC with zero (number of words to be inserted) and ESC with the count of the number of words to delete from the display list.
2	Load the GPU EPWC with the word count of the first item at which the effect of deletion is to be viewed. The previous examples (SELECT and HIT modes) both will provide a PWC which may be used as the word count at which the deletion is started.
3	Run the GPU with Edit-Enable bit ON [CTL (14)].
4	If desired, ESC may be adjusted under operator control from any input device such as control dials, etc.

4-29   Insertion

An example of a program ("INSERT") which could be used to view the effect of a possible insertion in an existing display is presented in Table 4-7. The program, in this case, will display to a viewer the apparent results of inserting seven words at PWC 134 of the picture being displayed. The words to be inserted are stored at CPU location BUF.

Table 4-7. Sequence for Sample INSERT Program

Step	Instruction
1	Load address of insert sequence BUF into GPU EA register.
2	Load length of insertion, 7, into GPU edit insert word count register EIC.
3	Load GPU edit skip word count register ESC with zero (no deletion).
4	Load GPU edit start picture word count register EPWC with 134, the word count of the designated item at which insertion is to be tried.
5	Run GPU with Edit-enable bit ON [CTL (14].

Substitution of arbitrary word sequences may be tried and viewed with the two EDIT-AIDS described thus far. Adjustments to existing elements being displayed during insertion may be made tentatively for viewing satisfaction. This simplifies "trial changes" by not having to locate the original elements and not having to expand or contract their memory requrements when trial changes vary the desired element length. The words in BUF (the trial word sequence) may be added to, deleted, or have their values altered to the desires of the monitoring operator through use of input devices. In this manner, the operator may continuously view his adjustment of any parameter being adjusted, i.e. , line length, line slope, circle radius, end-point position, sub-object selection, sub-object scale, rotation, etc.

4-30.   ADDRESSING

This section describes the GPU addressing functions used to locate Objects and to obtain any of their parameters through reference (Refr) addressing.

4-31   DISPLAY PICTURE DESCRIPTION

All display descriptions processed by the Display Processor consist of Objects which are located through the host computer Directory table. Each object may contain calls to lower level sub-objects which, in turn, may themselves consist of lower level sub-objects, etc. Figure 4-26 illustrates an example of three levels of objects which may be processed for display refresh in the following sequence:
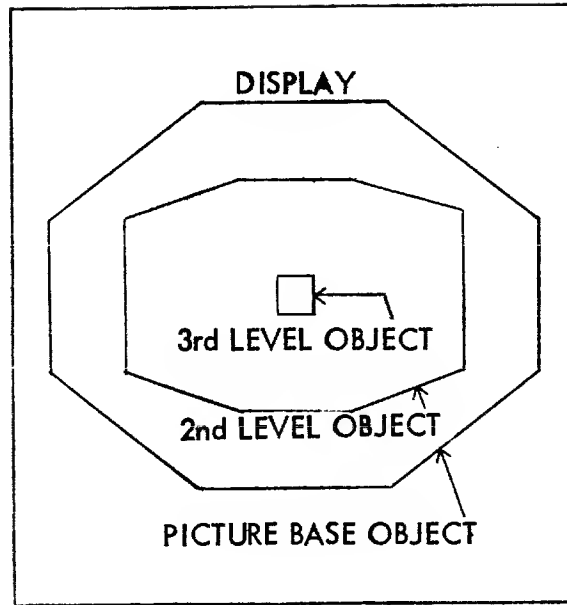
Figure 4-26. Display Consisting of Three Object Levels

a) The picture base object number (PBO), and the absolute address of the Directory in which it is located, are specified by the program as the data base start point. The GPU uses PBO as the initial current object number (OBN) to access the associated object list and process it to start generation of the refresh list which it outputs to the RBU.

b) The second-level object is then called, processed, and added to the refresh list.

c) The third-level object is then called, processed, and added to the refresh list.

d) A return is made from the third-level object, through the second-level object to the calling picture base object.

e) A return at the end of the picture base object list then terminates the update pass.

The updated refresh list is now in the RBU and available to the DCU to update the CRT display.

4-32. INITIAL ADDRESSING OF PICTURE BASE OBJECT

The following examples of addressing contain descriptions relative to GPU hardware functions which are included here to supplement the addressing information. Figure 4-27 illustrates the addressing of the picture base object and the generation of the GPU fetch address which is used to access the first instruction to process. In this example, it is assumed that a Directory address of 1000 and a PBO number of 100 are specified by the program as the data base start point. A brief description of the operation follows:

a) The directory address and the picture base object number are loaded via programmed outputs into the DIR and PBO registers respectively. (The stack base address and the stack limit must also be loaded via programmed outputs into the STB and SLM registers).

b) A programmed output is issued to the GPU command register to initialize processing of the picture base object.

c) The PBO number (the count of the number of words from the start of the directory to the picture base object address) is loaded into the GPU OBN register. The OBN (in this case 100, refer to Figure 4-27) is then used to index into the directory to access OBA (at location 1100 in this case).

d) The absolute address of the object list, OBA, is read from the directory and loaded into the GPU OBA register.

e) The first word of each object contains the count of the number of words from the start of the object to the first instruction in the list (in this case 10). This count is added ot OBA and placed in the GPU instruction address register IA.

f) The IA contents are then used as the fetch address to access the first instruction in the base object list (note that the base object's LINKS and LOCAL-OWN values are skipped at this time).

Processing of the picture base object is now started.
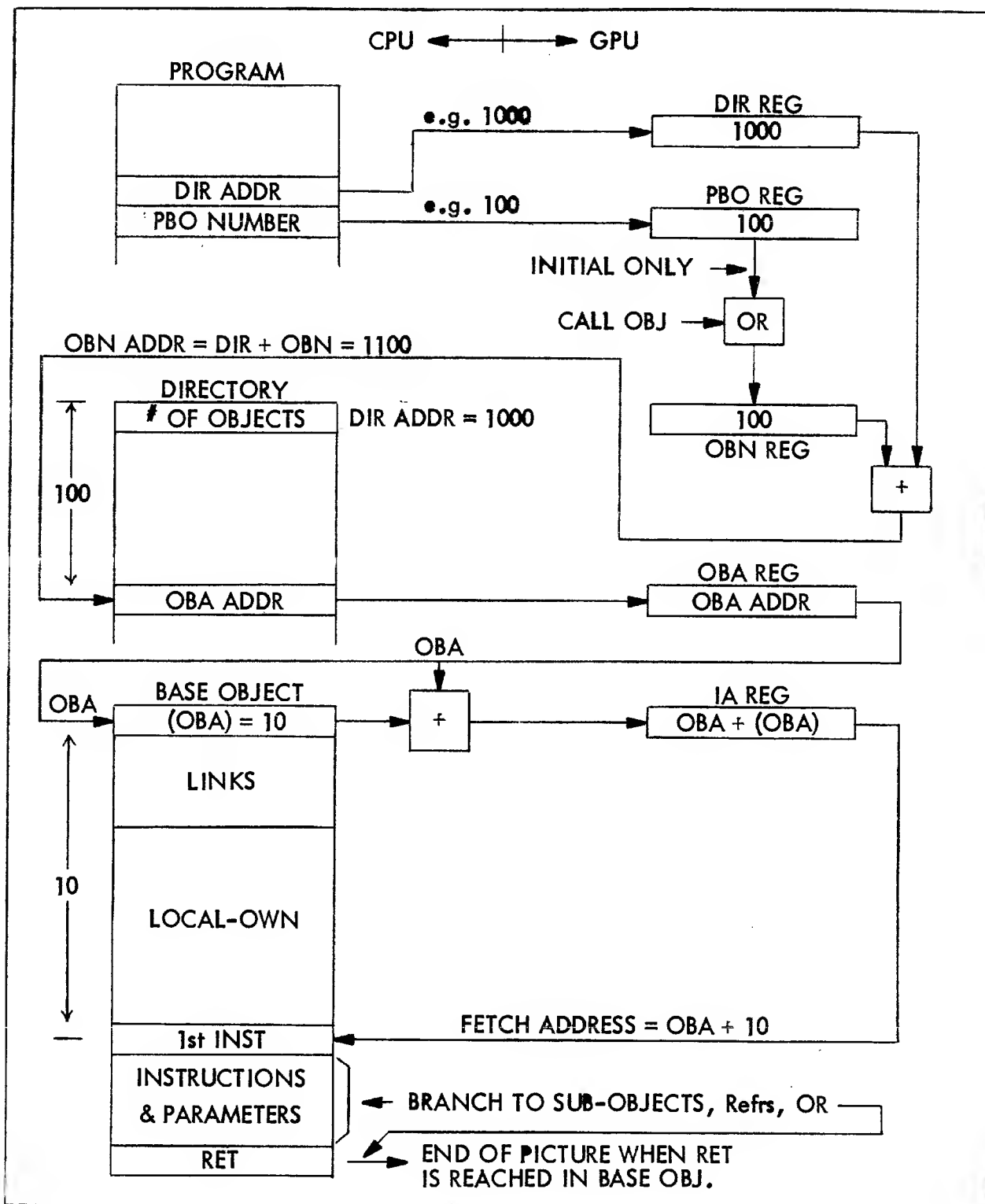
Figure 4-27. Initial Addressing of Base Object for GPU Processing

4-33. OBJECT LIST CONTENTS

Figure 4-27 also illustrates that the object list consists of links, local-own values, instructions with associated parameters and is terminated with a RETURN instruction. A brief description of the object list's contents and usage follows:

Object Base Address - The contents of this location, the first word in the object list, contains a count of the number of words to the first instruction in the list. This value is added to the object base address to access the first instruction in the object to start processing.

Links - The values in these locations are the indices to the directory entries (for calling objects as described in paragraph 3-13 or indices to data areas referenced externally. These locations are addressed by external or external-indexed Refrs in instruction word lists.

Local-Own - These locations contain local constants or own-variables and are addressed by local indexed Refrs in the instruction word lists.

Instructions and Parameters - This area contains instructions for GPU processing which may be followed either by immediate data values or by addressed references for locating variable values.

RET Instruction - All object lists are terminated with a RET instruction to either return to a higher-level "calling" object or to terminate the current process-update pass. If the RET is in the highest-level object (PBO), the current process-update is terminated; if not, a return is made from this object list to the "calling" higher-level object by returning to the instruction (+1) at which a call was made from that object. This is accomplished as follows:

a) The current "called" object's local stack is first cleared of all temporaries and MARKS (stacked by PUSH, MPUSH, and GMARK) and nested values (stacked by NEST and NESTI). The temporaries are discarded while the nested values are restored to their originating registers.

b) The addressing environment of the "calling" object is then restored by unstacking that object's local stack base address (SA), the element number (ELN), the object address (OBA), the object number (OBN), and the Resume Address (IA + 1). Unstacking of all temporaries and nested values in this local stack area continues up to, but not including, the first mark encountered. The nested values are again restored to their originating registers.

c) The instruction following the original CALL in the "calling" object list is then fetched to continue processing that object's list.

Figure 4-28. Referenced Variable Values or Indirect Addressing

4-39
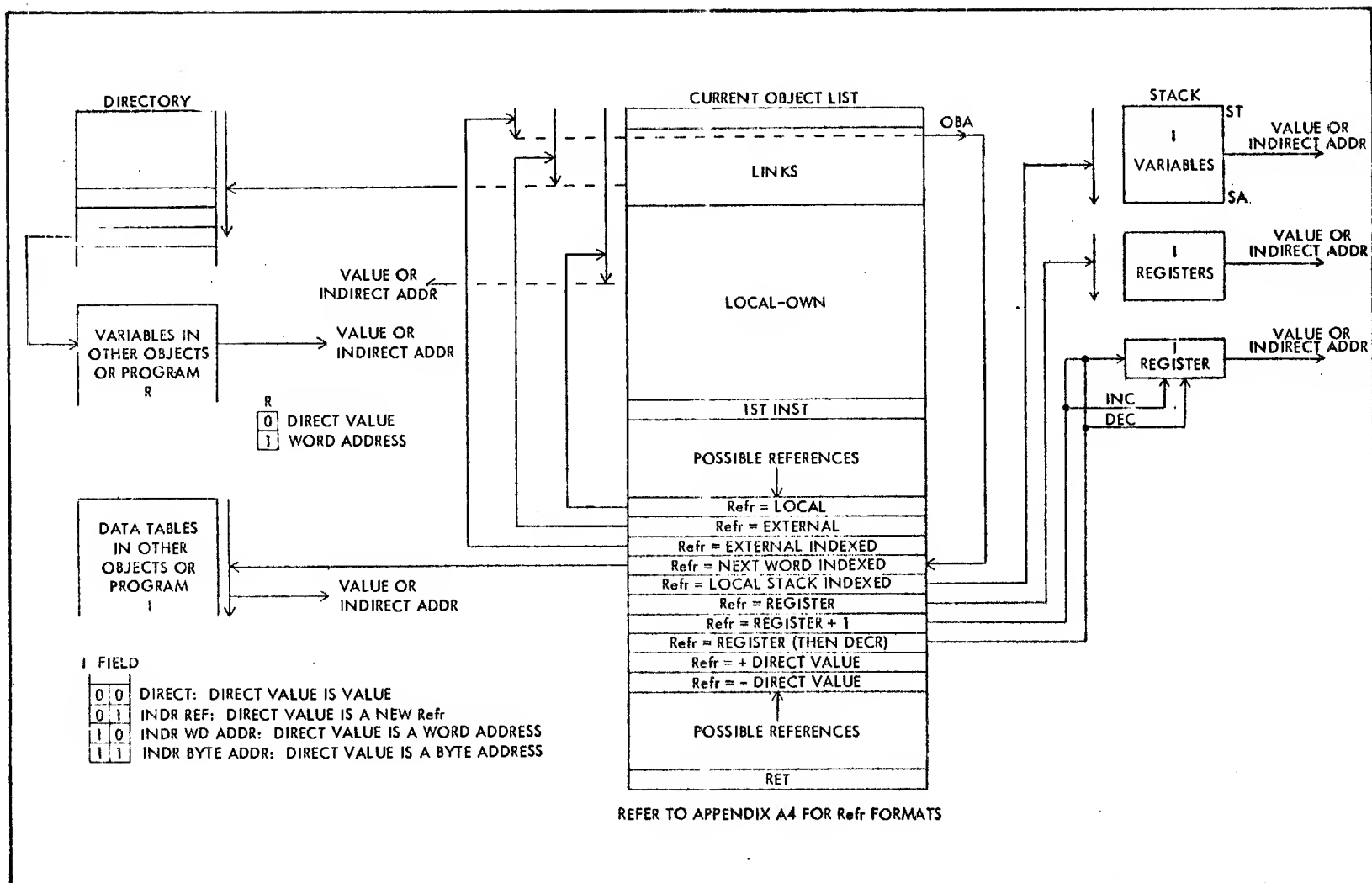
4-34.   REFERENCE ADDRESSING

Most instructions require one or more values to specify their operations (i.e., co-
ordinates for line generation, angle for a rotation operator, etc.). These values may
be either provided with each instruction as immediate values, or be located through
use of addressed references. Figure 4-28 illustrates the reference addressing tech-
nique. GPU instructions which may be followed by Refrs are as follows:

a)   LOAD Instructions - The Refrs are used to specify the destination addresses
     into which values are loaded and also each source address from which data
     is to be obtained.

b)   NEST Instructions - The Refrs are used to address arguments required to
     generate transforms which are to be stored in the registers specified by the
     Register Set Code.

c)   STACK Instructions - The Refrs are used to specify source addresses from
     which values are to be moved into the stack.

d)   LINES Instructions - The Refrs, determined by the LF field, are used to
     specify either or both the length or address of the list for the list of values
     of element generating parameters. They may also be used to locate individual
     coordinate values if specified by the DF field.

e)   TEXT Instructions - The Refrs are used to locate page-position parameters
     (if specified by "page control" field PG), to extract fields to establish ROT,
     FNT and SZ, and to locate individual characters is specified by the DF field.

f)   ARITH Instructions - The Refrs are used to locate the parameters on which
     the operations are to be performed and to specify the destination addresses to
     which the results are moved.

g)   Single Element Instructions - The Refrs specify the locations of the parameters
     needed to generate circles, arcs, rectangles, and cubic curve segments.

4-35.   STACK OPERATIONS

Stack operations are performed when any of the following seven instructions are
fetched: CALL, RET, NEST, POP, PUSH, GMARK, and MPUSH. Brief descriptions
of each of the operations are provided in the following paragraphs.

4-36. CALL INSTRUCTION EXAMPLE

The CALL instructions (refer to paragraph 3-13) effect a "call" to a lower-level sub-object by stacking a return instruction sequence in the "calling" object's local stack and then establishing a new stack area to be used by the "called" sub-object. Figure 4-27 illustrates an example of initial addressing of the first object in a picture description for GPU processing. The example in Figure 4-29 illustrates the method by which a CALL from a current object list (in this case ABC, object number 3) stores that object's addressing environment in the stack (Resume Address, OBN, OBA, ELN and SA which will be needed for an eventual return to ABC) and then sets up addressing to access the "called" sub-object list (DEF, object number 5). After stacking the re-store sequence, the local stack base register SA is set to object ABC's local stack top, thus establishing a fresh, empty, stack area for the called object named "DEF."

Each stack location is reserved or released under control of STK which operates as follows:

> Increment then store for a "write & reserve" (PUSH) operation.
> Read then decrement for a "read & release" (POP) operation.

The example in Figure 4-29 illustrates the address and stack configuration at the time the CALL DEF instruction is received (Figure 4-29A) and the result of the CALL operation (Figure 4-29B). The top section of the example shows that the current object (ABC) has been addressed by OBA, the instruction register IA is at address (b), the current element count in the object list is 12, and that ABC data (temporaries and/or nested values) have been stacked in ABC's local stack. When the CALL DEF instruction is encountered, three <u>basic</u> operations are performed:

a) Object ABC's addressing environment (Reseme Address, OBN, OBA, ELN and SA) are stacked for access at a later time to return to ABC (STK increments before each store).

b) The local stack base address SA is set to STK to provide fresh stack space for lower-level object DEF.

c) Object DEF's addressing environment is then set up to start processing DEF.

The environment for DEF is set up as follows:

a) IA at address (b) picks up OBJN (identified at LINK number 2).

b) The contents of LINK 2 (a count of directory address +5 in this case) are loaded in OBN to access DEF address, new OBA, from the directory.
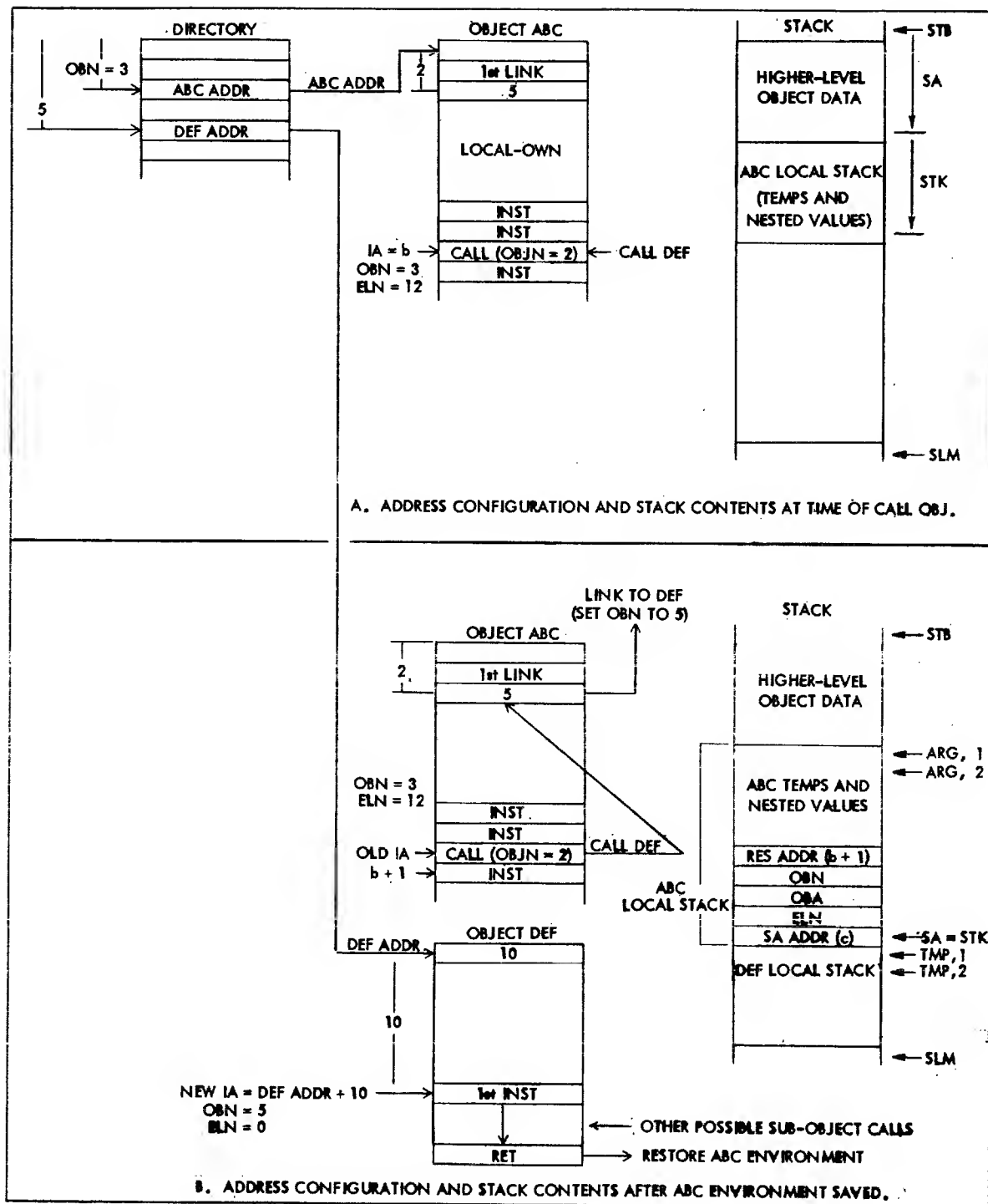
Figure 4-29. Call Object Instruction Example

c) The OBA count of 10 is added to the OBA address and loaded into IA to access the first instruction in DEF. The element count register ELN is also set to zero to prepare counting the elements in object DEF.

During the processing of DEF, a CALL to a lower-level object may be encountered, in which case the DEF addressing environment would be stacked in the DEF local stack and then the lower-level object would be accessed. However, in this example, it is assumed that no CALL instruction is encountered, and it is desired to return to object ABC. When processing of DEF is completed and the RET instruction is reached, the ABC environment is restored as follows:

a) The DEF local stack area is cleared by reading each location and decrementing STK after each read.

b) When STK finally reaches the top-of-stack for ABC (determined by STK equaling SA) the next five words (ABC environment) are restored. The base address of ABC's local stack is restored to SA and the four following words are restored to their appropriate registers. However, the RET instruction is not terminated at this time; temporaries and nested values in ABC's local stack continue to be cleared by reading and decrementing STK until a GMARK is encountered or until ABC's local stack becomes empty. At this time, processing of object ABC resumes.

4-37. SUMMARY OF STACK INSTRUCTIONS

Although there are only four stack-type instructions, the CALL, RET, and NEST instructions also perform stack operations as previously described. The following descriptions provide a summary of the operations performed by each instruction:

GMARK - This stack instruction places a one-word stack marker in stack. The mark is used to protect pushed and nested items across following calls and remains in effect until a POP instruction is encountered or the object which issued the GMARK issues a RET.

PUSH - This stack instruction (refer to paragraph 3-14) puts user-specified temporaries and arguments, followed by a control word, in the stack where they are then available for reference addressing. Stack area requirements are therefore n words +1. The temporaries remain in the stack until they are discarded upon a return to this local stack level (unless protected by a GMARK).

MPUSH - This stack instruction is identical with the PUSH instruction except that a stack marker is put in stack before the temporaries and control word. This requires a stack area of n words ÷2. The stack marker protects previously stacked items when a RET affecting this stack area is encountered, however, the items pushed by MPUSH are discarded by RET.

POP - This stack instruction is used to remove all temporaries (PUSH or MPUSH) and nested values (NEST) up to and including the first GMARK (or the GMARK produced by MPUSH) encountered in local stack.

4-38.   OBJECT LEVEL TRANSFORMATIONS

ELEMENT PROCESSING

$$
\begin{bmatrix} 1 & 0 & 0 & OD[X] \\ 0 & 1 & 0 & OD[Y] \\ 0 & 0 & 1 & OD[Z] \\ 0 & 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} R[XX] & R[XY] & R[XZ] & 0 \\ R[YX] & R[YY] & R[YZ] & 0 \\ R[ZX] & R[ZY] & R[ZZ] & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} OS & 0 & 0 & 0 \\ 0 & OS & 0 & 0 \\ 0 & 0 & OS & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} X \\ Y \\ Z \\ DS \end{bmatrix} \longrightarrow
$$

Displacement   Rotation   Scaling   Element

Window Transformation
Viewing Transformation
Object Generation

INITIALIZATION

	Object	Window	View
NPIC → GPU[CMD.] :	$1 \rightarrow DS$	$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \rightarrow \overline{WS}$	$1 \rightarrow PS$
	$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \rightarrow \overline{OD}$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix} \rightarrow \overline{WC}$	$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \rightarrow \overline{PD}$
	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \rightarrow [R]$	$1 \rightarrow WNZ$	
	$1 \rightarrow OS$		

EFFECT OF NEST INSTRUCTIONS ON OBJECT TRANSFORM PARAMETERS

NOS(S): Stack OS; $\quad S \times OS \rightarrow OS$

NODX(DX): Stack $\vec{OD}$; $\quad \vec{OD} + \left\{ OS \times [R] \times \begin{bmatrix} DX \\ 0 \\ 0 \end{bmatrix} \right\} \rightarrow \vec{OD}$

NODY(DY): Stack $\vec{OD}$; $\quad \vec{OD} + \left\{ OS \times [R] \times \begin{bmatrix} 0 \\ DY \\ 0 \end{bmatrix} \right\} \rightarrow \vec{OD}$

NODZ(DZ): Stack $\vec{OD}$; $\quad \vec{OD} + \left\{ OS \times [R] \times \begin{bmatrix} 0 \\ 0 \\ DZ \end{bmatrix} \right\} \rightarrow \vec{OD}$

NRX(RX): Stack [R]; $\quad [R] \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos RX & -\sin RX \\ 0 & \sin RX & \cos RX \end{bmatrix} \rightarrow [R]$

NRY(RY): Stack [R]; $\quad [R] \times \begin{bmatrix} \cos RY & 0 & \sin RY \\ 0 & 1 & 0 \\ -\sin RY & 0 & \cos RY \end{bmatrix} \rightarrow [R]$

NRZ(RZ): Stack [R]; $\quad [R] \times \begin{bmatrix} \cos RZ & -\sin RZ & 0 \\ \sin RZ & \cos RZ & 0 \\ 0 & 0 & 1 \end{bmatrix} \rightarrow [R]$

NOSXY(S,DX,DY): Stack OS,$\vec{OD}$; $\quad \vec{OD} + OS \times [R] \times (DX\ DY\ 0)^T \rightarrow \vec{OD}$; $\quad S \times OS \rightarrow OS$

NOSXYZ(S,DX,DY,DZ): Stack OS,$\vec{OD}$; $\quad \vec{OD} + OS \times [R] \times (DX\ DY\ DZ)^T \rightarrow \vec{OD}$; $\quad S \times OS \rightarrow OS$

NODXY(DX,DY): Stack $\vec{OD}$; $\quad \vec{OD} + OS \times [R] \times (DX\ DY\ 0)^T \rightarrow \vec{OD}$

NODXYZ(DX,DY,DZ): Stack $\vec{OD}$; $\quad \vec{OD} + OS \times [R] \times (DX\ DY\ DZ)^T \rightarrow \vec{OD}$

NORXYZ(RX,RY,RZ): Stack [R]; $\quad [R] \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos RX & -\sin RX \\ 0 & \sin RX & \cos RX \end{bmatrix} \begin{bmatrix} \cos RY & 0 & \sin RY \\ 0 & 1 & 0 \\ -\sin RY & 0 & \cos RY \end{bmatrix} \begin{bmatrix} \cos RZ & -\sin RZ & 0 \\ \sin RZ & \cos RZ & 0 \\ 0 & 0 & 1 \end{bmatrix} \rightarrow [R]$

NORZYZ(R1,RY,R3): Stack [R]; $\quad [R] \times \begin{bmatrix} \cos R1 & -\sin R1 & 0 \\ \sin R1 & \cos R1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos RY & 0 & \sin RY \\ 0 & 1 & 0 \\ -\sin RY & 0 & \cos RY \end{bmatrix} \begin{bmatrix} \cos R3 & -\sin R3 & 0 \\ \sin R3 & \cos R3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \rightarrow [R]$

STACK CONTENTS AFTER NEST (Local Stack initially has "L" entries)

Refr ARG Index	Stack OS	Stack $\overline{OD}$	Stack [R]	Stack OS, $\overline{OD}$
L + 1	OS	OD[Z]	R[ZZ]	OS
L + 2	2101	OD[Y]	R[ZY]	OD[Z]
L + 3		OD[X]	R[ZX]	OD[Y]
L + 4		2203	R[YZ]	OD[X]
L + 5			R[YY]	2104
L + 6			R[YX]	
L + 7			R[XZ]	
L + 8			R[XY]	
L + 9			R[XX]	
L + A			4409	

SECTION V

DISPLAY CONTROLLER HARDWARE DESCRIPTION

5-1.   DISPLAY REGISTERS

The standard display controller system (excluding the Display Processor) contains 16 addressable registers that can be subdivided into four categories as shown in Table 5-1. Additional registers can be provided as options. Appendix D1 lists all registers (both standard and optional), register formats, and the method of accessing these registers. The following descriptions of register formats and functions are limited to the 16 standard registers.

Table 5-1. Display Controller Registers (Standard), Station #1
(Addresses are provided in hexadecimal)

Control	Status	Instruction	Data	
DCU. (100)	DCUST. (102)	DCUI. (105)	SPX.	(10C)
VGU. (120)			SPY.	(10D)
MCU. (128)			SPZ.	(10E)
FGU. (130)			NAME.	(10F)
			XREG.	(124)
			YREG.	(125)
			ZREG.	(126)
			IREG.	(127)
			CHAR.	(131)
			CHSC.	(137)

5-2. CONTROL REGISTERS

Prior to drawing a display on the CRT screen, the DCU, VGU, MCU and FGU control registers must contain the desired display parameters (i.e., frame rate, frame mode, line type, character font and orientation, monitor selection, etc.). A description of each of the four control registers follows.

5-3. DCU CONTROL REGISTER (DCU.)

This 16-bit register (refer to Appendix D1) may be loaded by a LOAD instruction in the refresh list or by a programmed output from the GPU. The register contents are used to start and stop display refresh, enable display interrupts, acknowledge display interrupts, and to establish the frame rate and the frame mode. Operation of the register fields are described below.

START/STOP Field			–Controls CRT refresh as follows:	
00	01	02	–Start/Stop field bits	
0	0	0	–NOOP	
0	0	1	–NOOP	
0	1	0	–STOP;	resets START and RUN flip-flops to inhibit refresh list data transfer from RBU to DCU.
0	1	1	–START;	sets START and RUN to permit memory data requests by the DCU.
1	0	0	–Resume;	causes the DCU to restart data transfer from the RBU as specified by the frame mode (or to continue refresh at point it was interrupted).
1	0	1		
1	1	1	–Not permitted.	

ENSET, CLK, DEV, EOL, HLT*					–The clock, device, end-of-list, and interrupt-on-HALT enable bits are controlled as follows:
03	04	05*	06	07	–Interrupt enable bits.
1	1	1	1	1	–Set interrupt enable bits.
0	0	0	0	0	–No change.
0	1	1	1	1	–Reset interrupt enable bits.

ACK   –Bit 08   –Acknowledges display system interrupts.

*Pick window or light pen only.

FRAME MODE Field   -Controls CRT refresh as follows:

09	10	11		
			-Frame Mode field bits	
0	0	0	-NOOP	-No change.
0	0	1	-CUTOFF	-Frame clock cuts off picture and restarts refresh.
0	1	0	-ALL	-No cutoff, restart refresh at first frame clock following end of picture.
0	1	1	-CONTINUOUS	-No cutoff, restart refresh immediately after picture end.
1	0	0	-CLK 120Hz	-No cutoff, restart refresh at next 120-Hz clock.*
1	0	1	-FRM	-No cutoff, restart refresh and display single frame.
1	1	0	-OFF	-No auto refresh, finish picture if refreshing and don't restart.
1	1	1	-EXT	-External clock cuts off picture and restarts refresh.

*100 Hz if 50HZ input power

FRAME RATE Field   -Controls CRT refresh as follows:

12	13	14	15		
				-Frame Rate field bits	
0	0	0	0	-No change.	
0	0	0	1	-120	120 frames/second
0	0	1	0	-120/2	60 " "
0	0	1	1	-120/3	40 " "
0	1	0	0	-120/4	30 " "
0	1	0	1	-120/5	24 " "
0	1	1	0	-120/6	20 " "
0	1	1	1	-120/7	17 " "
1	0	0	0	-120/8	15 " "
1	0	0	1	-120/9	13 " "
1	0	1	0	-120/10	12 " "
1	0	1	1	-120/11	11 " "
1	1	0	0	-120/12	10 " "
1	1	0	1	-120/13	9 " "
1	1	1	0	-120/14	8.5 " "
1	1	1	1	-120/15	8 " "

NOTE: The frame rates listed above are based on the use of 60-Hz input power. When 50-Hz power is used, the highest frame rate is 100 frames/second, and all other rates are divided down proportionally.

5-4.   VGU CONTROL REGISTER (VGU.)

The VGU Control Register (see hex address 120 in Appendix D1), containing only
3 bits of data, may be loaded by a LOAD or VECTOR instruction in the refresh list
or by the host computer during a programmed output. Register contents are used to
specify the type of line to be drawn on the CRT screen as follows:

LINE TYPE Field   -Controls type of line to be drawn as follows:

09	10	11	
09	10	11	-Line Type field bits
0	0	0	-No change.
0	0	1	-Draw solid lines.
C	1	0	-Draw long dashes.
0	1	1	-Draw short dashes.
1	0	0	-Draw long/short dashes.
1	0	1	-Draw long/short/short dashes.
1	1	0	-Point mode, unblank at end of vector move.
1	1	1	-Not used.

5-5.   MCU CONTROL REGISTER (MCU.)

The MCU Control Register (see hex address 128 in Appendix D1), containing 9 bits
of data, can be loaded by a LOAD instruction in the refresh list. Contents of this
register are used to select one of four display monitors (up to 8 optional).

MS ENABLE Field	MONITOR SELECT Field								-Selects monitors to be unblanked
00	08	09	10	11	12	13	14	15	
0	-	-	-	-	-	-	-	-	-No change to blanking scheme
1	1	0	0	0	0	0	0	0	-Enable unblank of Monitor 1, blank of Monitors 2 through 8
1	0	1	0	0	0	0	0	0	-Enable unblank of Monitor 2, blank of Monitors 1,3 through 8.
1	0	0	0	0	0	0	0	1	-Enable unblank of Monitor 8 blank of Monitors 1 through 7
1	1	1	1	1	0	0	0	0	-Enable unblank of Monitors 1 through 4, blank of Monitors 5 through 8

5-6.   FGU CONTROL REGISTER (FGU.)

The FGU Control Register (see hex address 130 of Appendix D1), containing 5 bits of data may be loaded by either a LOAD or CHARACTER instruction in the display list. The register contents are used to control the 90 degree character rotation feature and type of font as follows:

ROTATION Field   –Controls 90° rotation of characters as follows:

09	10	11	
			–Rotation field bits
0	0	0	–No change.
0	0	1	–No change.
0	1	0	–No change.
0	1	1	–No change.
1	0	0	–Normal (no rotation).
1	0	1	–90° CCW rotation.
1	1	0	–180° rotation.
1	1	1	–270° CCW rotation.

SLANT Field   –Controls character font as follows:

12	13	
		–Slant field bits
0	0	–No change.
0	1	–Select normal font.
1	1	–Select slanted font (right-slanted 26 degrees).

5-7. DISPLAY CONTROLLER STATUS REGISTER (DCUST.)

The Display Controller Status Register (see hex code 102 of Appendix D1), containing 14 bits of data, stores the status of the DCU and can be read only. A description of the status bits follows.

Bit	Function	
2	RUN	-The DCU is in the RUN mode.
3	2nd BYTE	-Indicates the second byte of a 16-bit word.
		Example: Y data in an incremental vector word or the second character in a character word.
4	CLK	-Clock interrupt bit (120Hz)
5	DEV	-Device interrupt bit (Light Pen or Pick Window)
6	EOL	-End-of-list interrupt bit.
7	HLT	-Interrupt bit for Instruction Control Halt and Interrupt.
8-11	HIT1-4	-Hit/Select option bits.
12-15	PEN SW	-Pen Switch option bits.

5-8.   <u>DISPLAY CONTROLLER INSTRUCTION REGISTER (DCUI)</u>

The Display Controller Instruction Register (see hex address 105 in Appendix D1) contains the instruction word of the current display refresh list. This specifies the operation to be performed by the system. The DCUI Register can be loaded only by an instruction word in the refresh list. However, its contents may be read by the host computer during a programmed input.

Four types of instructions can be specified by the DCUI Register: (a) CONTROL, (b) LOAD, (c) VECTOR, and (d) CHARACTER. Additional information concerning these four types of instructions can be found in the following paragraphs. The OP CODE Field (bits 00 and 01) is shown below. Other fields are illustrated in Figure 5-1. Bit 14 is always a 1; bit 15 is always a 0.

OP CODE Field		-Specifies one of four types of instruction words in current display refresh list.
00	01	-Op Code field bits
0	0	-Control Instruction
0	1	-Load Instruction
1	0	-Vector Instruction
1	1	-Character Instruction
Other Fields		-see Figure 5-1

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	
0	0	Cntrl Type												1	0	CONTROL INST

0	1		COUNT		DESTINATION		1	0	LOAD INST
DATA									
DATA									LOAD DATA LIST
DATA									

1	0		3D	BLNK	COLOR	LINE TYPE	0	0	1	0	VECTOR ABS INST
±	COORDINATE						REG		OF		ABS DATA

1	0		3D	BLNK	COLOR	LINE TYPE	0	1	1	0	VECTOR REL INST
±	Δ COORDINATE						REG		OF		REL DATA

1	0		3DI	BLNK	COLOR	LINE TYPE	1	S	1	0	VECTOR INC INST
±	Δ X COORDINATE		±	Δ Y COORDINATE							2D INC DATA
±	Δ Z COORDINATE										3D INC DATA

1	1		3D	BLNK	COLOR	ROTATE	SLANT	1	0	CHARACTER INST
CHARACTER #1				CHARACTER #2						TEXT/CONTROL

Figure 5-1. Display Refresh Instructions and Data Format

5-9.   CONTROL INSTRUCTION

The CONTROL instruction (see example in Figure 5-1) functions to halt the display controller with or without an interrupt. The CONTROL TYPE Field is shown below. In the case where both bit 2 and bit 3 of the CONTROL TYPE Field are set, three events occur in sequence: (a) the display controller halts, (b) an interrupt is generated if bit 7 of the DCU CONTROL REGISTER (hex 100 in Appendix D1) is set, and (c) bit 7 of the DCUST DISPLAY CONTROLLER STATUS REGISTER (hex 102 in Appendix D1) is set to signify that the interrupt has been generated.

CONTROL TYPE Field		-Specifies the type of control instruction.
02	03	-Control Type field bits
0	0	-NOOP
0	1	-Not used.
1	0	-HALT
1	1	-HALT and Interrupt.

5-10. LOAD INSTRUCTION

The LOAD instruction (see second example in Figure 5-1) works in conjunction with a list of data words to load the display registers specified by the DESTINATION and COUNT fields. As illustrated in the second example of Figure 5-1, the data words must immediately follow the LOAD instruction, and the number of data words must agree with the COUNT field. Notice that a COUNT field of 0000 signifies one register to be loaded, 0001 signifies two registers, etc.

DESTINATION Field	-Contains the address of the first destination register (the first register to be loaded).
COUNT Field	-Contains the count of the number of registers to be loaded successively.
03 04 05 06	-Count field bits
0 0 0 0	-Load one register (specified by the DESTINATION address).
0 0 0 1	-Load two registers in succession (starting at DESTINATION address).
0 0 1 0	-Load three registers in succession.
1 1 1 1	-Load 16 registers in succession.

5-11.   VECTOR INSTRUCTION

The VECTOR instruction permits vectors to be drawn. As shown in Figure 5-1, the type of vector draw depends upon the VECTOR MODE Field (bits 12 and 13 of the VECTOR instruction. The four possible vector modes are as follows: (a) Vector Absolute, (b) Vector Relative, (c) Vector Incremental and (d) Vector Incremental with Smoothing. An absolute vector is drawn by using coordinates that are referenced to a fixed origin, the center of the monitor (0, 0). A relative vector is drawn with coordinates that are referenced to the end of the previous vector, using a full 12 bits for any single coordinate. A vector incremental is simply a shortened version of a relative vector, using 8 bits to describe any coordinate, and thereby permitting both X and Y coordinates to be specified in a single instruction (also Z coordinate in 3-DI). Finally, an incremental vector with smoothing resembles the normal incremental vector draw except that an added analog look-ahead feature provides rounded transitions rather than sharp ones. These VECTOR instructions are briefly described in subsequent paragraphs.

VECTOR MODE Field   -Controls mode of vector draw or move.

12	13	
		-Vector Mode field bits
0	0	-Vector absolute
0	1	-Vector relative
1	0	-Vector incremental
1	1	-Vector incremental with smoothing

5-12.   Vector Absolute

The VECTOR ABSOLUTE instruction causes the data word following the 16-bit instruction word to be loaded into the register specified by the REG field, with an operation performed by the VGU in accordance with the OF field. The vector is formed by an absolute coordinate, referenced to the center of the monitor. Figure 5-2 illustrates the VECTOR ABSOLUTE refresh instruction and the associated VECTOR ABSOLUTE data.

	00	01	02	03	04 05	06 07 08	09 10 11	12	13	14	15
VECTOR ABS INSTRUCTION	1	0		3D	BLNK	COLOR	LINE TYPE	0	0	1	0
ABS DATA	±	COORDINATE						REG		OF	

Figure 5-2.   Vector Absolute Instruction and Data Format

3-D Field*   -Specifies either 2-D or 3-D vector

03   -3-D field bit
 0   -2-D vector
 1   -3-D vector

BLNK Field   -Controls blinking of displayed element(s) for all vector instructions as follows:

04 05   -Blink field bits
 0   0   -No change.
 0   1   -No change.
 1   0   -Disable blink mode.
 1   1   -Enable blink mode (2 Hz rate).

*The 3-D Field is incorporated for use with the VECTOR INCREMENTAL and CHARACTER instructions. This field has no function in either VECTOR ABSOLUTE or VECTOR RELATIVE instructions.

COLOR Field* -Selects color of display and velocity of beam

06	07	08	-Color field bits
0	0	0	-Black and White - Velocity 0 (fastest)
0	0	1	-Yellow - Velocity 2
0	1	0	-Orange - Velocity 3
0	1	1	-Red - Velocity 4 (slowest)
1	0	0	-Green - Velocity 1
1	0	1	-Amber - Velocity 3
1	1	0	-reserved
1	1	1	-reserved

*Effective April, 1978

LINE TYPE Field -Controls type of line to be drawn.

09	10	11	-Line Type field bits
0	0	0	-No change.
0	0	1	-Draw solid lines.
0	1	0	-Draw long dashes.
0	1	1	-Draw short dashes.
1	0	0	-Draw long/short dashes.
1	0	1	-Draw long/short/short dashes.
1	1	0	-Point mode, unblank at end of vector move.
1	1	1	-Not used.

REG Field -Specifies the register (in both the DCU and VGU) to be loaded.

12	13	-Register field of vector data word.
0	0	-Load data into XREG.
0	1	-Load data into YREG.
1	0	-Load data into ZREG.
1	1	-Load data into INTENSITY REG (refer to Appendix E1 for GSX mode)

OF Field -Specifies operation to be performed by the VGU.

14	15	-Operation field bits of vector data word.
0	0	-Load register specified by REG field.
0	1	-Load specified register and move.
1	0	-Not permitted (This code would identify the word as an instruction).
1	1	-Load specified register and draw.

5-13.   Vector Relative

The VECTOR RELATIVE instruction causes the data word following the 16-bit in-
struction word to be <u>added</u> to the contents of the register specified by the REG field,
with an operation performed by the VGU, in accordance with the OF field. The vec-
tor is formed by a relative coordinate, referenced to the end of the previous vector.
Figure 5-3 illustrates the VECTOR RELATIVE refresh instruction and the associa-
ted VECTOR RELATIVE data. Refer to paragraph 5-12 for a description of the
fields.

	00	01	02	03	04 05	06 07 08	09 10 11	12	13	14	15
VECTOR REL INST	1	0		3D	BLNK	COLOR	LINE TYPE	0	1	1	0
REL DATA	±				Δ COORDINATE				REG	OF	

Figure 5-3. Vector Relative Instruction and Data Format

5-14.   Vector Incremental

The VECTOR INCREMENTAL instruction generates a relative vector with 8-bit rel-
ative coordinates. Figure 5-4 shows the VECTOR INCREMENTAL refresh instruc-
tion and the associated data. The 3-DI field is described below. The S field (bit 13)
is the smoothing bit: 0= no smoothing, 1 = smoothing. Other fields are identical to
those in paragraph 5-12.

	00	01	02	03	04 05	06 07 08	09 10 11	12	13	14	15
VECTOR INC INST	1	0		3DI	BLNK	COLOR	LINE TYPE	1	S	1	0
2-D INC DATA	±	Δ X COORDINATE				±	Δ Y COORDINATE				
3-D INC DATA	±	Δ Z COORDINATE									

Figure 5-4. Vector Incremental Instruction

3-DI Field

03   -3-DI field bit

2-DI   -This instruction causes the two bytes of each following data word (delta X and delta Y, reference Figure 5-1) to be multiplied by 16 and added to the current values of the XREG and YREG. A vector draw is then made to that new endpoint. Each successive data word is handled in the same manner until a data word consisting of hex 0001 is received to terminate the 2-DI instruction

3-DI   -This instruction causes the delta X, delta Y, and delta Z values in the next two data words (reference Figure 5-1) to be multiplied by 16 and added to the current values of the XREG, YREG, and ZREG. A vector draw is then made to that new endpoint. The terminate code (hex 0001) is the same as in 2-DI instruction above.

S Field

13   -S field bit
0   -No smoothing
1   -Smoothing made

5-15.   CHARACTER INSTRUCTION

The CHARACTER instruction, followed by two-byte character words (refer to Figure 5-5) causes any of the 96 ASCII characters to be drawn or causes the control circuitry to perform the control functions described in Table 5-2. The ROTATE and SLANT fields are forwarded to the FGU register; the COLOR field is sent to the MCU. When the instruction is received, all subsequent words are interpreted as character codes (refer to Appendix E1) until either a terminate code 9C or a GSX code is received. Table 5-2 describes the control codes recieved by the DCU from the display refresh list.

	00	01	02	03	04 05	06 07 08	09 10 11	12 13	14	15
CHAR INST	1	1		3D	BLNK	COLOR	ROTATE	SLANT	1	0
TEXT/CONTROL	CHARACTER #1					CHARACTER #2				

Figure 5-5. Character Instruction and Text Control

3-D   –Select 2-D or 3-D operation.

03   –3-D field bit.
0   –2-D operation.
1   –3-D operation.

BLNK Field   –Controls blinking of displayed element(s) for all vector instructions
              as follows:

04 05   –Blink field bits.
0   0   –No change.
0   1   –No change.
1   0   –Disable blink mode.
1   1   –Enable blink mode (2-Hz rate).

CHARACTER

Color Field   –Select color of display and velocity of beam.

06	07	08		
			–Color field bits.	
0	0	0	–Black and White	– Velocity 0 (fastest).
0	0	1	–Yellow	– Velocity 2
0	1	0	–Orange	– Velocity 3
0	1	1	–Red	– Velocity 4 (slowest).
1	0	0	–Green	– Velocity 1
1	0	1	–Amber	– Velocity 3
1	1	0	–reserved	
1	1	1	–reserved	

ROTATION Field   –Controls 90° roation of characters as follows:

09	10	11	
			–Rotation field bits.
0	0	0	–No change.
0	0	1	–No change.
0	1	0	–No change.
0	1	1	–No change.
1	0	0	–Norman (no rotation).
1	0	1	–90° CCW rotation.
1	1	0	–180° rotation.
1	1	1	–270° rotation.

SLANT Field   –Controls character font as follows:

12	13	
		–Slant field bits.
0	0	–No change.
0	1	–Select normal font.
1	1	–Select slanted font (right-slanted 26 degrees).

Table 5-2. Display Controller Character Control Codes

Code (HEX)	Mnemonic	Description
9A	SUB	Subscript (1/4 line feed, 2/3 font size). The DCU transfers this code to the FGU which decodes it to perform the the subscript functions (the VGU performs no repositioning for this code).
9B	RSZ	Reduce character size (2/3 font size). The DCU transfers this code to the FGU which decodes it to reduce character size. This code remains active until an RLC code is received to restore font size to normal.
9C	FSX	Derived from the ASCII FS (field separator). The DCU decodes this control code to terminate the character string and interpret the following word as a new instruction.
9D	GSX	Derived from the ASCII mnemonic GS (group separator). The DCU decodes this control code and interprets the next word as position data (same as vector data format) to signal the VGU to perform a vector move to a new character position. The OF field of 01 or 11 (load and move or load and draw) in the position data word then returns the DCU to the character string (reference also Appendix E1).
9E	RLC	Restore line coordinates. The DCU transfers this code to the FGU which decodes it to negate the SUB, SUP, and RSZ functions.
9F	CUR	Cursor. This code is decoded by the DCU to enter the character blink mode (4 Hz) and also transfers the code to the FGU which performs the cursor draw. The VGU performs no character spacing after the draw.
99	SUP	Superscript (1/4 reverse line feed. 2/3 font size). The DCU transfers this code to the FGU which decodes it to perform the superscript functions (the VGU performs no repositioning for this code).
08	BS	Back space. The DCU decodes this control code and sends vector move data to the VGU which performs the backspace function.

5-16. DISPLAY CONTROLLER DATA REGISTERS

Display Controller Data Registers (see Table 5-1) include the following: (a) character spacing (SPX., SPY., SPZ.), (b) vector endpoint coordinates (XREG., YREG., ZREG.), (c) display intensity (IREG.), (d) character codes (CHAR.), (e) character scale (CHSC.), and (f) identification (NAME). These registers are described below.

5-17. CHARACTER SPACING (SPX., SPY. SPZ.)

These 16-bit registers (addresses 10C, 10D, and 10E in Appendix D1) are loaded during a LOAD instruction from the display refresh list. They store the transformed character spacing coordinates required by the VGU to perform vector moves between character draws from one character position to the next. Character Spacing Registers can be read by the CPU during a programmed input.

5-18. VECTOR ENDPOINT COORDINATES (XREG., YREG. ZREG.)

The 15-bit registers (sign bit plus 11 bits of magnitude, smoothing bit and operation field OF) are loaded from the display refresh list during vector absolute instructions, or modified by the refresh list during vector relative and incremental instructions. The contents of these registers specify the current transformed endpoint coordinate of the vector draw or move to be performed by the VGU. The three registers may also be loaded by a LOAD instruction in the refresh list and read by the CPU during a programmed output.

5-19. DISPLAY INTENSITY (IREG.)

This 8-bit register provides the facility to vary the display intensity in 255 discrete steps from minimum (80_{16}) to maximum ($7F_{16}$). The register may either be loaded by a LOAD or VECTOR ABSOLUTE instruction in the display refresh list or be modified by a VECTOR RELATIVE instruction. It then holds the intensity value to be used by the VGU during the current vector/character draw. It may be read by the CPU during a programmed input.

5-20. CHARACTER REGISTER (CHAR.)

This 8-bit register, loaded by either a LOAD instruction or a character word in the display list, holds the character code for the current character draw (or current character control code). Its contents are not available to the CPU.

5-21. CHARACTER SCALE REGISTER (CHSC.)

This 8-bit register (sign bit plus 7 bits of magnitude) enables the FGU to scale character size to any of the four sizes specified by program. Paragraph 5-26 describes character scaling and provides examples of the scaling effects on characters and character space. The register may be loaded either by a LOAD instruction in the refresh list or by a data word in a vector instruction if specified by a GSX. Its contents are not available to the CPU.

5-22. NAME REGISTER (NAME.)

This 16-bit register may be loaded by either a LOAD instruction in the display refresh list or by a programmed output. Its contents may also be read by the CPU during a programmed input.

5-23. FONT GENERATOR UNIT CHARACTERISTICS

The FGU utilizes a stroke technique whereby characters are formed by drawing a series of straight-line segments. Through use of a 512-word ROM, all characters illustrated in Appendix E1 may be drawn by specifying character codes 21 through 7F and 9F (HEX). Each character draw is performed by successively accessing a series of 16-bit words from the ROM and performing a segment draw (or move) as each word is processed. The fields associated with each word are presented in Table 5-3 for use in explanation of character space, size, drawing slopes, scaling and rotation.

Table 5-3. FGU ROM Word Format

Field Bits	Field
0-4	X coordinate value
5-9	Y coordinate value
10-12	Line slope
13	Intensity (draw in place of move)
14	Terminate character draw
15	Skip terminate

The 5-bit X and Y coordinate values are limited to a range of 0-31 providing a character space (refer to Figure 5-6) comprised of a 32 by 32 matrix consisting of 1024 programmable points that can be used for character draws. It should be noted, however, that each bit in the FGU X and Y coordinates represents 10 bits in the display screen X and Y coordinate system with the result that the character space occupies a total of 310 by 310 raster units on the display screen. The upper case "A" character, drawn near the center of the character space in Figure 5-6, shows an aspect ratio of 18/12 or 3/2. The "A" character therefore occupies 180 by 120 raster units within the 310 by 310 raster unit character space. Assuming a full scale viewing area of 13 by 14 inches (3.4 mils per bit on the display screen) this results in an upper-case, unscaled, character size of 0.62 inch high by 0.408 inch wide. Since the FGU uses the stroke technique for drawing, even at this large size the characters remain crisp with constant intensity.

5-24.   CHARACTER DRAWS

Figure 5-6 shows a character space with X0, Y0 in the lower left corner and X31, Y31 in the upper right corner. When the previous character draw was completed, the FGU returned to the "parking position" (X10, Y9), the point at which the following character draw (or move) will start. The draw of the "A" character is performed using the following steps:

a) A draw is first made from X10, Y9 to X16, Y27.

b) The second draw is then made from X16, Y27 to X22, Y9.

c) A move is made from X22, Y9 to X20, Y15 (unintensified).

d) The last draw is made from X20, Y15 to X12, Y15.

e) A move is then made to the parking position.

It can be seen in the above that a slope was drawn (3Y, 1X) in both steps (a) and (b). The restrictions on drawing slopes are described in the following paragraph.
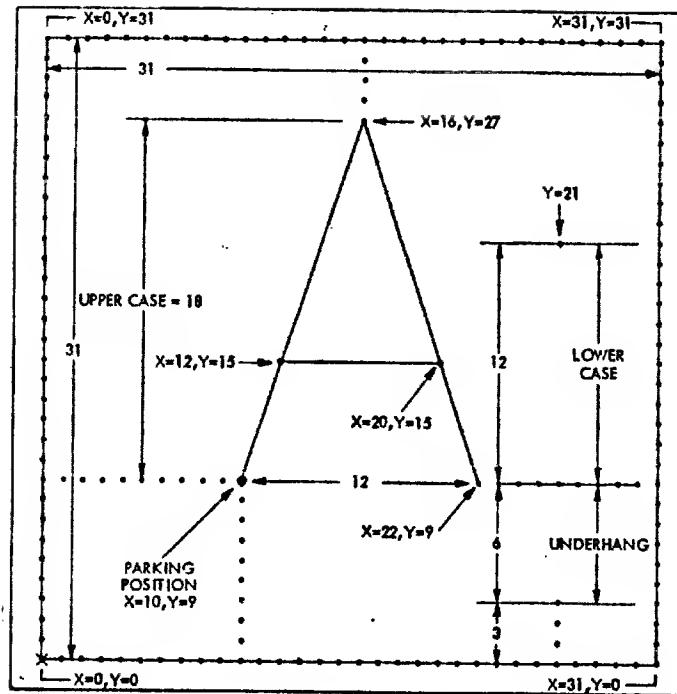
Figure 5-6. Character Space and Character Position

5-25. LINE SEGMENT SLOPES

Line segments used during the character draws are restricted to a total of nine slopes (refer to Figure 5-7). To ensure a straight-line draw from the current beam position to a new endpoint, two parameters must be specified, i.e., the X-Y coordinate position of the new endpoint and the slope of the draw to that point. For example, a line is drawn from point X9, Y3 to point X14, Y27; i.e., the beam moves 8 X-units and 24 Y-units. To ensure that the line is straight, the Y velocity must be three times the X velocity, or slope 3Y/1X as shown in Figure 5-7. When drawing horizontal or vertical lines, and also when performing moves between segments, the highest drawing speed (3Y/3X is used. An example of the use of slopes when drawing the upper-case "A" is presented in Table 5-4.

Table 5-4. Parameters Used For Drawing "A"

Operation	Y	X	Slope	Intensity	Terminate	Draw
a) At Parking Position	9	10	———	———	———	•
b) 1st draw	27	16	18/6 = 3Y/1X	1	0	↗
c) 2nd draw	9	22	18/6 = 3Y/1X	1	0	↘
d) Move	15	20	3Y/3X	0	0	↖
e) Last draw	15	12	3Y/3X	1	1	←
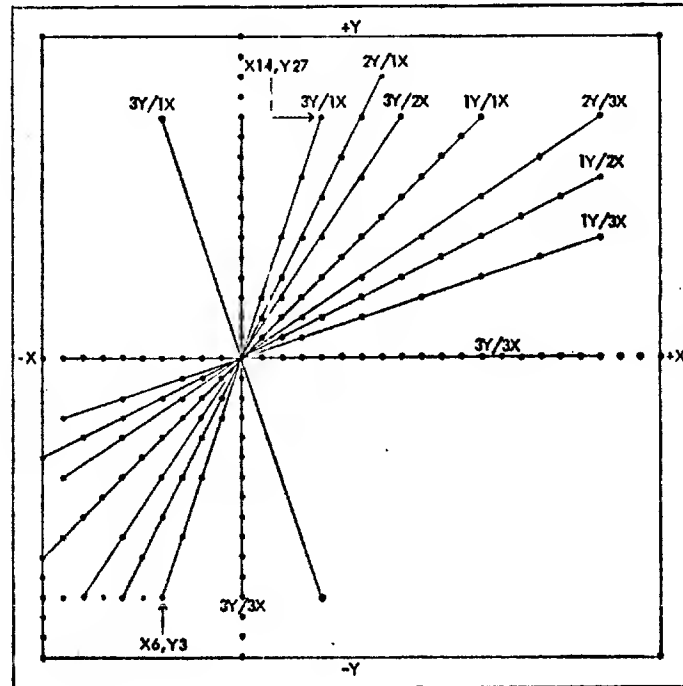
Figure 5-7. Permissible Line Slopes

5-23

5-26. CHARACTER SCALING

Although the GPU may specify any one of four standard character size scaling, (refer to Table 1-4), the FGU contains an 8-bit character scale register which enables characters to be scaled from a positive full scale value to a negative full scale value. The register contents are in 2's complement notation for negative numbers (bit 00 is the sign bit) and control scaling as follows (hexadecimal notation is used):

7F = positive full scale value
40 = positive 1/2 full size
01 = positive 1/27 full size
00 = zero size
FF = negative 1/128 full size
C0 = negative 1/2 full size
80 = negative full scale value
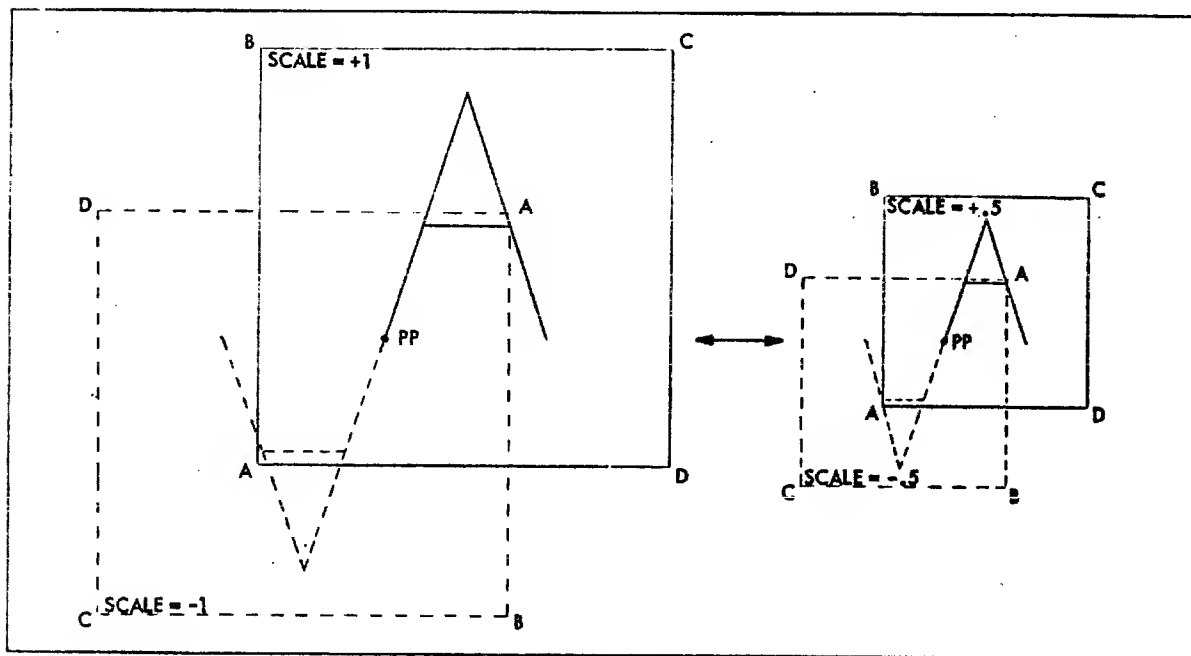
Figure 5-8. Effects of Character Scaling

Figure 5-8 shows the beam parking position (PP), i.e., the X-Y coordinate point within the character space at which the beam rests prior to starting the character draw. This point remains stationary when scaling is used while the character and the space which it occupies are scaled proportionally. When negative scale values are used (negative values are never specified by the GPU), the image wraps around on itself (and also the character space) and becomes a mirror image of the original. The parking position is also the center point for the standard 90° rotation feature.

5-27. CONTINUOUS CHARACTER ROTATION

The continuous character rotation feature provides for programmable, continuous rotation of characters about their parking positions. Since the program must provide X-Y coordinate data to the VGU for character position, spacing, etc., this feature required that the following requisites be satisfied by program:

    a)   The program must rotate the character space, line-feed, etc. X-Y coordinates proportionately and supply them to the VGU for page space rotation.

    b)   The program must specify the rotation transforms which are loaded into the FGU rotation registers (CR11, CR12, CR21, and CR22, refer to Appendix D1) for character rotation.

Figure 5-9 illustrates an example of (a) an untransformed character string, (b) rotation of the positioning coordinates (page space rotation), and (c) final rotation of characters within the rotated page space.

A description of Figure 5-9 follows:

5-9(a)   With no rotation, a character-space vector $(\overline{\text{SP}})$ equals (h, 0, 0) and a character-line-feed vector $(\overline{\text{LF}})$ equals (0, v, 0).

5-9(b)   To rotate the page space, the new character-space vector = (hx, hy, hz) and the new character-line-feed vector = (vx, vy, vz).

5-9(c)   To rotate the characters proportional to page space rotation, the FGU rotation registers are loaded as shown in Figure 5-9(c).

NO ROTATION
a

$\overline{SP} = (h, 0, 0)$
$\overline{LF} = (0, v, 0)$

PAGE ROTATION
b

$\overline{SP} = (hx, hy, 0)$
$\overline{LF} = (vx, vy, 0)$

FINAL ROTATION
c

CR11 = hx/h
CR12 = vx/v
CR21 = hy/h
CR22 = vy/v
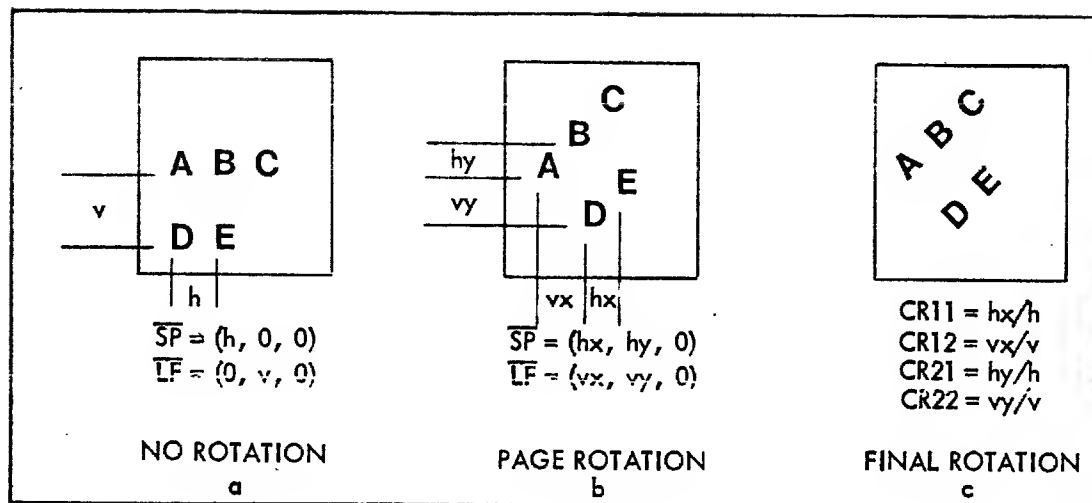
Figure 5-9.   2-D Page Space and Character Rotation

SECTION VI

GLOSSARY OF TERMS

6-1.   INTRODUCTION

Table 6-1 provides a list of special terms and descriptions in the context in which they are used in this document. To aid the user in programming, some descriptions have been expanded to illustrate the techniques used to generate display descriptions; e. g. , Windowing, Zooming, etc.

Table 6-1. Special Terms and Descriptions

Absolute Vector

    A vector whose endpoint coordinates are specified with respect to the zero coordinate position at the center of the CRT screen and $\pm$ full scale at the screen edges (or the local object's number space if transformed with respect to the screen).

Alphanumeric Keyboard

    An optional entry device for manually inputting characters and control codes into the display system.

Blinking

    A standard feature whereby the program may highlight any part of the display by blinking the desired portion at a 2-Hz rate. This feature is also used in the Hit/Select option to highlight a specific element in the display. When the Cursor character is displayed it is always automatically blinked at a 4-Hz rate.

Character Set

    The character set consists of the standard 96 ASCII character set of which 95 codes provide character draws. The remaining code is used as a character space code. In addition, 16 control codes are used for control of the character displays (the cursor control code also provides a draw of the Cursor character).

Table 6-1. Special Terms and Descriptions (Continued)

Character Spacing

When the FGU completes a character draw, a move (character space) is automatically made by the VGU to the next character position (except in the case of a Cursor character draw).

Circles

When specified by the program, circles are drawn on the display by placing the VGU in the "smoothing mode" and drawing a series of short incremental vectors.

Circular Arcs

Similar to "Circles" above, circular arcs are also drawn by the VGU in the "smoothing" mode.

Clipping

Clipping is the procedure whereby the data extracted for viewing in the window area is limited to the extraction boundaries specified by the windowing parameters and also to the boundaries of the display screen. This is accomplished by clipping and discarding unseen elements and recomputing the endpoints of partially visible elements (see Windowing).

Control Codes (Character)

Seventeen character codes are used for control purposes and are either not displayed or they modify the display of characters which they control. The control functions are as follows: Backspace, Horizontal Tab, Line Feed, Vertical Tab, Form Feed, New Line, Space, Reverse Line Feed, Half Line Feed, Half Reverse Line Feed, Superscript, Subscript, Reduce Size, Terminate, Restore Line Coordinates, Blinking Cursor, and Delete.

Cursor

A blinking cursor symbol (⌐___⌐) which can be drawn at a "current" character position. It may be used by an editing program in conjunction with the keyboard to select a character position at which a deletion and/or insertion is to be performed.

Table 6-1. Special Terms and Descriptions (Continued)

Data Scale

A standard feature whereby all untransformed, incoming coordinate data from the CPU may be scaled. This allows any composite display to be scaled down until all its transformed components lie within the 16-bit dynamic range of the display processor.

Data Tablet

The data tablet is an optional graphic input device with an X-Y coordinate grid. Information is entered into the display system from the tablet via a penlike stylus.

Default Values

When the Display is initialized by the GPU, the following default values are set:
Frames per second to 40Hz
Frame mode to ALL
Vector draw velocity to maximum (if opted)
Select Monitor #1
Start DCU

Depth Cueing

A technique in which the Z-axis coordinate is transformed and used to shade the image intensity on the display to provide the 2-D projected display with a 3-D viewing effect.

Depth Perspective

A technique whereby display elements parallel to the section walls of a view port may be varied from parallel to 90° divergent as the perspective viewpoint (PDZ) is varied from zero to maximum. This then provides depth perspective of the extracted data portion displayed in the view port (see Windowing).

Table 6-1. Special Terms and Descriptions (Continued)

Directory

An area set aside in CPU memory which contains the addresses of all Objects contained in a picture description and also the addresses of any externally accessable data or data tables.

Displacement

The displacement feature permits moving any individual object intact, with respect to the three axes of the display, by specifying X, Y, and Z object displacement values. The entire displayed picture may also be moved with respect to the X-Y axes of the display screen by specifying X-Y picture displacement values.

Display Controller

The display controller, consisting of the DCU, VGU, FGU, and MCU, provides all signals required by the display monitor to generate and control the displayed picture.

Display Control Unit (DCU)

The DCU processes both the refresh list received from the RBU, and the inputs from the Light Pen and Hit/Select options, to direct all display functions. It also responds to programmed I/O instructions to provide data transfers to/from addressed registers in the DCU, VGU, FGU, MCU, Light Pen Interface, and the Hit/Select logic.

Display List

The display list, consisting of user instructions and associated data, is generated by the application program and stored in CPU memory for subsequent processing by the display processor to generate the display refresh list. The refresh list is stored in the RBU and used by the DCU to update the CRT display.

Display Monitor

The display monitor, comprised of a CRT, major and minor deflection channels, and a video channel, displays the picture generated from the refresh list. The unit utilizes an electromagnetic deflection system which moves the electron beam across the face of the

Table 6-1. Special Terms and Descriptions (Continued)

CRT, in response to either the major deflection inputs from the VGU or to the minor X-Y inputs from the FGU. Signal inputs from the MCU control both the electron beam unblanking and the screen intensity level.

Display Processing

The display processor, consisting of the GPU and associated RBU(s), receives the display list from the CPU, performs all transformations specified by the program, and generates refresh lists comprised of 2-D picture descriptions suitable for display. The GPU stores the refresh list in the RBU where it is accessed by the DCU during a display refresh cycle. The processor also provides aids for interaction and on-line display modification.

Display Refresh

Display refresh is the electronic regeneration of the picture displayed on the CRT each time the DCU processes the refresh list. Each regenerated picture is called a "frame" and the rate at which refresh occurs is called the frame rate (e.g., 40 frames/second).

DMA Read

DMA read operations are performed when the GPU requests data from addressed CPU memory locations.

DMA Write

DMA write operations are performed when the GPU signals the CPU memory that it wishes to write data into an addressed memory location.

Edit Aid Option

This option provides hardware for interactive (tentative) deletion, insertion, and adjustment of displayed elements while utilizing display visual feedback to observe their effects. This permits altering the display by substitution of arbitrary word sequences in the display list with trial word sequences which may be adjusted until visual satisfaction is achieved.

Table 6-1. Special Terms and Descriptions (Continued)

Element

Elements are the building blocks for objects and are considered as the smallest visible portions of objects displayed on the CRT, such as individual lines or vectors, characters, etc.

Element X, Y, Z Coordinates

The X, Y, and Z coordinates of elements specify the endpoint location to which the CRT beam is to be moved during a vector, text, curve, or other element generation.

Element Incremental Values

Incremental values of elements are the X, Y, and Z values which are to be added to the current X, Y, and Z element coordinates when performing incremental draws or moves.

Font Generator Unit (FGU)

The FGU generates the analog deflection voltages, in response to character codes received from the DCU, which are applied to the minor deflection channels (X-Y) of the display monitor in order to perform the specified character draws. It also contains logic circuitry to control character size, rotation, slant, etc.

Frame Clock

The frame clock, synchronized with the 60-Hz input power, is programmable from 8 Hz to 120 Hz and establishes the frame rate at which the display is refreshed when in the ALL, CUTOFF, and CLK frame modes. When using 50-Hz input power the frame clock may be programmed from 6.67 to 100 Hz.

Frame Modes

Any of seven frame modes may be program selected to control the mode of display refresh. This is accomplished by specifying the conditions under which the DCU accesses the refresh list from the RBU.

Table 6-1. Special Terms and Descriptions (Continued)

Frame Rate (see Frame Clock)

Highlighting (see Blinking)

Hit/Select Option

This option provides the facility to highlight elements for identification when specified programatically. It also provides a means of recording the word position in the display list for an element which is "Hit" via the light pen or pick options.

Incremental Vectors

Incremental vectors provide a means of economizing on data storage and increasing the rate of output and display (number of vectors/second). Two dimensional incremental vectors (2-DI) require only one data word for both the X and Y incremental values while a 3-DI vector requires two words for the X, Y, and Z values.

Instructions (Display List)

The control codes generated by the application program which together with coordinate and character data, constitute a picture description that can be processed by the GPU.

Instructions (Micro)

The display processor firmware within the GPU which, under control of the display list instructions, generates the refresh list required by the display controller.

Instructions (Refresh List)

The control codes generated by the display processor which, together with associated coordinate and character data, are stored in the RBU and accessed by the DCU to generate the displayed picture.

Intensity Modulation

Intensity modulation is the name given to the intensity depth cueing transformation in all 3-D systems. It shades the intensity of the final 2-D displayed picture in relation to its Z-axis values to give a 3-dimensional viewing effect.

Table 6-1. Special Terms and Descriptions (Continued)

Interactive Devices

Six optional devices may be connected to the display system: an Alpha-
numeric Keyboard, a Light Pen, a Data Tablet, a Joy Stick, Function
Switches and Control Dials.

Interrupt Channel

This channel is used by the display system and device response inter-
rupts to activate computer programs.

Joy Stick

The joy stick is an optional mechanical device used to enter X, Y, and
Z coordinate information into the display system.

Light Pen

The light pen is an optional device which generates a "Hit" when its
photocell detects the visible element at which it is pointing on the dis-
play screen.

Monitor Control Unit (MCU)

The MCU is used to select the desired monitor to be used for display
and to generate the final intensity and unblank signals for the monitor.
In addition, when different vector draw velocities are specified by the
program, the monitors not intended to be driven at the new draw rate
are automatically blanked.

Object (see also Sub-object)

An object is a set of display instructions and associated data, re-
ceived by the GPU from the CPU, which form the building blocks for
the final displayed picture. Each object may consist of an element
or groups of elements.

Object Displacement

All objects may be moved intact with respect to the 3 axes by specify-
ing X, Y, and Z object displacement values.

Table 6-1. Special Terms and Descriptions (Continued)

Object Transformation

The transformation of an object is described by its scaling, displacement and rotation.

Object Rotation

An object may be rotated about any angle in all three axes on the display by specifying the X, Y, and Z rotation values.

Object Scaling

An object may be scaled in all axes simultaneously by specifying an object scale value.

Perspective (see Depth Perspective)

Pick Window

This option defines a pick window, under control of an interactive device, which may be moved about on the CRT screen and used for item selection. The selected item may be highlighted for identification.

Picture Base Object

The picture base object is the highest-level or "main" object in the CPU directory. It is the first object received by the display processor for processing and may call other objects as "sub-objects." The calls continue to be performed until all objects have been processed and the complete refresh list for a display frame has been generated.

Picture Displacement (PDX, PDY, PDI)

This displacement permits part or all of the displayed picture to be moved intact with respect to the two axes on the display by specifying X and Y displacement values.

Picture Scale (PS)

Tne entire displayed picture may be controlled in size by specifying the desired picture scale value. This function is especially useful in "zooming," or enlarging portions of the picture for closed examination (see Windowing and Zooming).

Table 6-1. Special Terms and Descriptions (Continued)

Picture Word Count (PWC)

Each word in the CPU data base is represented by a picture word count. The GPU uses this count to cause elements selected by the program to be highlighted and to notify the program of the PWC of an element selected by a device "Hit."

Programmed Input (PI)

The PI is a method whereby the contents of the display system registers are brought into the computer.

Programmed Output (PO)

The PO is the method whereby control and acknowledgement information is sent to the display system. It is also used to write data into addressed display system registers.

Random Scan

The display monitor uses the random scan method of controlling movement of the CRT electron beam. The beam is steered between two points on the display screen, a series of which form an image portion. The FGU also uses the random scan technique for character draws.

Refresh Buffer Unit (RBU)

The RBU is used to store the refresh list generated by the GPU. The standard RBU is an 8K word memory of 16 bits/word which optionally may be expanded in increments of 8K to a maximum of 32K words.

Refresh List

The refresh list is a list of display instructions and data accessed by the DCU from the RBU and used to display one frame on the CRT.

Relative Vectors

The endpoint coordinates of a relative vector are located with respect to the start point coordinates. In other words, relative vector data are specified in the form of increments that are added to or subtracted from the previous coordinate data values.

Table 6-1. Special Terms and Descriptions (Continued)

Resolution

Deflection resolution is defined as the number of addressable points per inch on the display screen. Example: 4096 bits per 14" = 293 addressable points per inch. Usable display resolution, however, is limited by the CRT spot size and becomes 1/20 mils = 50 lines/inch (20 mil tube) or 1/10 mils = 100 lines/inch (10 mil tube). Data resolution is the number of distinct values the data may assume. A 14-bit field has a resolution of 2^{14}.

Rotation

Using 3-dimensional rotation coordinates, any object may be rotated about any of the three axes.

Scaling

The scaling operation consists of changing the size of an image portion by multiplying each endpoint coordinate by the desired scale factor before processing. The scale factors are specified by the application program and are in the form of both object and picture scaling. In addition, characters may be specified as any of four standard text sizes defined by the GPU.

Small Element Discard

Small element discard is the feature whereby the output of successive display elements is suppressed when they would lie within a 13 mil distance (for a standard 13" by 14" viewing area) of the previously displayed element endpoint.

Smoothing

The smoothing mode permits use of a series of short incremental vectors to draw curved lines.

Stack (RAM)

The RAM stack permits several levels of sub-object nesting to be stored without accessing the CPU memory for environment saving and restoring.

Table 6-1. Special Terms and Descriptions (Continued)

Stack Base

The stack base is the lowest address of the stack area.

Stack Limit

The stack limit is the highest address of the stack area.

Stack Pointer

The stack pointer is a register containing the address of the current level top-of-stack. The pointer is normally decremented after a stack "read" and incremented before a stack "write."

Sub-Object (see also Object)

A sub-object is an object in the CPU display list which may be called by a higher level object. Calls may also be performed from a sub-object to a lower level sub-objects.

Transformation

Transformation is the process of modifying a display image by scaling, displacement and/or rotation.

Variable Velocity Control

This option provides the facility which permits the program to select drawing speeds on the CRT varying from 0.75 inch/μs maximum to 1 inch/500μs minimum. This allows several monitors with different sweep characteristics to be operated by one vector and/or font generator.

Vector Generator Unit (VGU)

The VGU receives digital inputs from the DCU and generates the analog voltages for the display monitor major deflection channels to draw solid or dashed lines between any two locations on the CRT or to place a point at any given position. It also performs the vector moves between character positions while the FGU is operating in the character draw mode.

Table 6-1. Special Terms and Descriptions (Continued)

Windowing

Windowing is the technique in which a portion of displayed data is extracted and presented for viewing within a designated area on the screen.

The location of the data portion to be extracted is specified by positioning its center, both horizontally and vertically, in the final transformed data space using window WCX and WCY values and establishing the front (near) Z cutoff plane using a WNZ value.

The size of the data portion to be extracted comprises a rectangular section about WCX and WCY out to $\pm$ WSX horizontally and $\pm$ WSY vertically.

The depth of the section is the portion of the data behind the near cutoff plane WNZ and may be adjusted in size from zero to maximum (meaning all data in the section behind WNZ) by specifying WSZ values from zero to full scale respectively.

The section walls will vary from parallel to 90° divergent as the perspective viewpoint PDZ is varied from zero to maximum.

The area on the display screen at which the extracted portion is to be viewed is established by specifying its location (PDX, PDY) and its size (PS). PDX and PDY range from plus to minus maximum which permits full scale (half screen) deflection both horizontally and vertically. Size PS may range from zero to maximum positive (full screen size).

Work Station

A work station as specified by the Devices STA# and the RBU/DCU STA# fields, in the GPU CMD. register, or by standard display address allocations (Appendix B1, C) is comprised of an RBU/DCU pair or a group of input devices.

Writable Control Store (WCS)

The WCS is an option which permits the user to modify, extend, or replace entirely both the firmware implementing the user instruction set and the constants and immediate values normally stored in ROM. The

Table 6-1. Special Terms and Descriptions (Continued)

standard GPU firmware, when unaltered by not incorporating this option, consists of a microprogram which is described thoroughly in the Display Processor Technical Manual.

Zooming

Zooming is the technique in which window-extracted data displayed on the screen is magnified or reduced to provide a "zooming-in" or "zooming-out" effect. This results when the picture size (PS) is greater than or less than the window size parameters (WSX, WSY) respectively (see Windowing).

APPENDICES

CONTENTS

*   These tables of hardware and device registers and hardware
decoding are provided for information purposes. Descriptions
of the register fields are provided in Section V.

USER INSTRUCTION SET

Operation	Octal Code	Decimal Code	Hex Code	Mnemonic	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
CONTROL OPS	00000	0000	0000	NOOP	0	0	0	0												
	10000	4096	1000	RETU	0	0	0	1	0	0										
	14000	6144	1800	RETZ	0	0	0	1	1	0										
	16000	7168	1C00	RETNZ	0	0	0	1	1	1										
	20000	8192	2000	GHALT	0	0	1	0												
	30000	12288	3000	BRKL	0	0	1	1	NUMBER OF WORDS											
DATA MOVE OPS	40000	16384	4000	LOAD	0	1	0	0	0	NUMBER OF VALUES										
	44000	18432	4800	LOADI	0	1	0	0	1	NUMBER OF VALUES										
	50000	20480	5000	NEST	0	1	0	1	0	REGISTER SET CODE										
	54000	22528	5800	NESTI	0	1	0	1	1	REGISTER SET CODE										
STACK OPS	60000	24576	6000	CALLU	0	1	1	0	0	OBJECT LINK INDEX										
	64000	26624	6800	CALLC	0	1	1	0	1	OBJECT LINK INDEX										
	70000	28672	7000	POP	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
	70000	28672	7000	PUSH	0	1	1	1	0	NUMBER OF VALUES										
	74000	30720	7800	GMARK	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
	74000	30720	7800	MPUSH	0	1	1	1	1	NUMBER OF VALUES										
ARITHMETIC OPS	100000	32768	8000	GADD	1	0	0	0	0								0	0	0	0
	100001	32769	8001	GSUB													0	0	0	1
	100002	32770	8002	GMPY													0	0	1	0
	100003	32771	8003	GDIV													0	0	1	1
	100004	32772	8004	GAND													0	1	0	0
	100005	32773	8005	GOR													0	1	0	1
	100006	32774	8006	GXOR													0	1	1	0
	100007	32775	8007	GSHFT					0								0	1	1	1
	104000	34816	8800	GADDI					1								0	0	0	0
	104001	34817	8801	GSUBI													0	0	0	1
	104002	34818	8802	GMPYI													0	0	1	0
	104003	34819	8803	GDIVI													0	0	1	1
	104004	34820	8804	GANDI													0	1	0	0
	104005	34821	8805	GORI													0	1	0	1
	104006	34822	8806	GXORI													0	1	1	0
	104007	34823	8807	GSHFTI	1	0	0	0	1								0	1	1	1
ELEMENT LIST OPS	130000	45056	B000	ARBI	1	0	1	1	0			NUMBER OF WORDS								
	134000	47104	B800	ARB	1	0	1	1	1			NUMBER OF WORDS								
	140000	49152	C000	LINES	1	1	0	0	LF		DF		BM		CLX		CLY		CLZ	
	160000	57344	E000	TEXT	1	1	1	0	LF		DF		PG		ROT		FNT		SZ	
SINGLE ELEMENT OPS	170000	61440	F000	CIRCLE	1	1	1	1	0								0	0	0	0
	170001	61441	F001	CCWARC													0	0	0	1
	170002	61442	F002	CWARC													0	0	1	0
	170003	61443	F003	RECT													0	0	1	1
	170004	61444	F004	CUBIC					0								0	1	0	0
	174000	63488	F800	CIRCL4					1								0	0	0	0
	174001	63489	F801	CCARC4													0	0	0	1
	174002	63490	F802	CWARC4													0	0	1	0
	174003	63491	F803	RECT4													0	0	1	1
	174004	63492	F804	CUBIC4	1	1	1	1	1								0	1	0	0

USER INSTRUCTION WORD FIELDS

Field	Octal Code	Hex Code	Mnemonic	Description	Refr Page
List Field (LF)	000	000	LFIT	Immediate list with terminate	3-23, 3-35
	2000	400	LFRT	Referenced list with terminate	3-24, 3-36
	4000	800	LFIC	Immediate list with count	3-24, 3-36
	6000	C00	LFRC	Referenced list with count	3-24, 3-37
Data Field (DF)	000	000	DFWD	Full word data format	3-25, 3-38
	400	100	DFBY	Byte data format	3-26, 3-39
	1000	200	DFB4, DF7B	LINES Byte/4 data format, TEXT 7-bit format	3-27, 3-40
	1400	300	DFRF	Referenced data format	3-27, 3-41
Beam Sequence Field (BM)	000	000	BMDJ	Disjoint line string	3-28
	100	040	BMJL	Joined line string	3-29
	200	080	BMHV	Horizontal/vertical run	3-30
	300	0C0	BMPT	Setpoint (move)	3-31
X Load Field (CLX)	000	000	CCX	Constant coordinate loading field	3-32
	020	010	CIX	Stepped coordinate loading field	3-32
	040	020	CAX	Absolute coordinate loading field	3-32
	060	030	CRX	Relative coordinate loading field	3-32
Y Load Field (CLY)	000	000	CCY	Constant coordinate loading field	3-33
	004	004	CIY	Stepped coordinate loading field	3-33
	010	008	CAY	Absolute coordinate loading field	3-33
	014	00C	CRY	Relative coordinate loading field	3-33
Z Load Field (CLZ)	000	000	CCZ	Constant coordinate loading field	3-34
	001	001	CIZ	Stepped coordinate loading field	3-34
	002	002	CAZ	Absolute coordinate loading field	3-34
	003	003	CRZ	Relative coordinate loading field	3-34
Page field (PG)	000	000	PGNC	Current page	3-42
	100	040	PGBM	Text block	3-43
	200	080	PG00	Absolute page	3-44
	300	0C0	PGXY	Positioned page	3-45
Rotation Field (ROT)	000	000	RONC	No change in character rotation	3-47
	020	010	RO00	Reset to zero degree character rotation	3-47
	040	020	ROPK	Referenced word rotation field	3-47
	060	030	RORF	Referenced right-justified rotation field	3-47
Font Field (FNT)	000	000	FNNC	No change in font	3-48
	004	004	FN00	Reset font to non-slanted characters	3-48
	010	008	FNPK	Referenced word font field	3-48
	014	00C	FNRF	Referenced right-justified font field	3-48
Size field (SZ)	000	000	SZNC	No change in character size	3-49
	001	001	SZ80	Reset to standard 80 characters/line size	3-49
	002	002	SZPK	Referenced word size field	3-49
	003	003	SZRF	Referenced right-justified size field	3-49

REGISTER SET CODES FOR NESTING VALUES

Code			Mnemonic	#ARGS	N/M	Description
DEC	HEX	OCT				
00	00	00	NOSXY	3	N	2D object scale, X, Y displacements (OS, ODX, ODY)
01	01	01	NOSXYZ	4	N	3D object scale, X, Y, Z displacements (OS, ODX, ODY, ODZ)
02	02	02	NODXY	2	N	2D object X, Y displacements (ODX, ODY)
03	03	03	NODXYZ	3	N	3D object X, Y, Z displacements (ODX, ODY, ODZ)
04	04	04	NORXYZ	3	N	3D rotation, Z then Y then X (RZ, RY, RX)
05	05	05	NORZYZ	3	N	3D rotation, Z then Y then Z (Euler angles) (RZ, RY, RZ)
06	06	06	NOS	1	N	Object scale (OS)
07	07	07	NODX	1	N	Object X displacement (ODX)
08	08	10	NODY	1	N	Object Y displacement (ODY)
09	09	11	NODZ	1	N	Object Z displacement (ODZ)
10	0A	12	NRX	1	N	X rotation value (RX)
11	0B	13	NRY	1	N	Y rotation value (RY)
12	0C	14	NRZ	1	N	Z rotation value (RZ)
13	0D	15	MPSIXY	4	M	Picture scale, intensity scale, X, Y displacements (PS, PDI, PDX, PDY)
14	0E	16	MWCXYS	5	M	Window center point, near Z, size X, size Y (WCX, WCY, WNZ, WSX, WSY)
15	0F	17	MPDXY	2	M	Picture X, Y displacements (PDX, PDY)
16	10	20	MWCXY	2	M	Window center point (WCX, WCY)
17	11	21	MPS	1	M	Picture scale (PS)
18	12	22	MPDX	1	M	Picture X displacement (PDX)
19	13	23	MPDY	1	M	Picture Y displacement (PDY)
20	14	24	MWCX	1	M	Window X center point (WCX)
21	15	25	MWCY	1	M	Window Y center point (WCY)
22	16	26	MWS	2	M	Window size (WSX, WSY)

NOTES: The above codes are used for NEST and NESTI instructions only.

#ARGS = The number of arguments used in the modifications.

M = The current values of the specified registers are first stacked and are then replaced with new argument values.

N = The current values of the specified registers are first stacked after which new values are composed (by applying the arguments to the current values) and loaded into the specified registers.

REFERENCE ADDRESSING

Group	Octal Code	Decimal Code	Hex Code	Mnemonic	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	Description
REF	00000	0000	0000	IMD+	0	0	0	+DIRECT VALUE													IMMEDIATE POSITIVE
REF	20000	8192	2000	DEV	0	0	1	0	0	0	I/O DEVICE ADDRESS										DEVICE
REF	24000	10240	2800	TMP	0	0	1	0	1	IND		0	STACK INDEX								TEMPS IN LOCAL STACK
REF	24400	10496	2900	ARG	0	0	1	0	1	IND		1	STACK INDEX								ARGS IN CALLER'S STACK
REF	30000	12288	3000	REG	0	0	1	1	0	IND		0	REGISTER								REGISTER
REF	30400	12544	3100	RGI	0	0	1	1	0	IND		1	REGISTER								REGISTER INCREMENTED
REF	34400	14592	3900	RGD	0	0	1	1	1	IND		1	REGISTER								REGISTER DECREMENTED
REF	40000	16384	4000	LOC	0	1	IND		OBJECT LOCAL STORAGE INDEX												LOCAL
REF	100000	32768	8000	EXI	1	0	IND		OBJECT LINK STORAGE INDEX												EXTERNAL INDEXED VALUE
REF	140000	49152	C000	EXV	1	1	0	R	OBJECT LINK STORAGE INDEX												EXTERNAL VALUE
REF	160000	57344	E000	IMD-	1	1	1	-DIRECT VALUE													IMMEDIATE NEGATIVE
IND (STK/REG)	00000	0000	0000							0	0		REGISTER								DIRECT VALUE
IND (STK/REG)	1000	512	0200	REF						0	1		REGISTER								NEW REFR (CANNOT BE TO EXI)
IND (STK/REG)	2000	1024	0400	INW						1	0		REGISTER								WORD ADDR OF VALUE
IND (STK/REG)	3000	1536	0600	INB						1	1		REGISTER								BYTE ADDR OF VALUE
IND (MEM)	00000	0000	0000				0	0	OBJECT LOCAL/OWN STORAGE INDEX												DIRECT VALUE
IND (MEM)	10000	4096	1000	REF			0	1	OBJECT LOCAL/OWN STORAGE INDEX												NEW REFR (CANNOT BE TO EXI)
IND (MEM)	20000	8192	2000	INW			1	0	OBJECT LOCAL/OWN STORAGE INDEX												WORD ADDR OF VALUE
IND (MEM)	30000	12288	3000	INB			1	1	OBJECT LOCAL/OWN STORAGE INDEX												BYTE ADDR OF VALUE
R Field	00000	0000	0000					0	OBJECT LINK STORAGE INDEX												DIRECT VALUE
R Field	10000	4096	1000	INW				1	OBJECT LINK STORAGE INDEX												WORD ADDR

Stack Contents After NEST (Local Stack initially has "L" entries)

Refr ARG Index	Stack OS	Stack $\overline{OD}$	Stack [R]	Stack OS, $\overline{OD}$
L + 1	OS	OD[Z]	R[ZZ]	OS
L + 2	2101	OD[Y]	R[ZY]	OD[Z]
L + 3		OD[X]	R[ZX]	OD[Y]
L + 4		2203	R[YZ]	OD[X]
L + 5			R[YY]	2104
L + 6			R[YX]	
L + 7			R[XZ]	
L + 8			R[XY]	
L + 9			R[XX]	
L + A			4409	

STANDARD 96 ASCII CHARACTER SET

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
								LSD →									
0										BS	HT	LF	VT	FF	NL		
1																	
2		Space	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3		0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4		@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5		P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	⌒	_
6		`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7		p	q	r	s	t	u	v	w	x	y	z	{	¦	}	~	⊠
8								RLF								HLF	HRL
9										SUP	SUB	RSZ	TRM			RLC	CUR

MSD (vertical axis, left side)

NOTE: CODES 21 THROUGH 7F AND 9F ARE PRINTABLE CHARACTERS.

USER CONTROL CODES

Code	Mnemonic	Description
08	BS	Backspace – causes the current character position to backspace one character position.
09	HT	Horizontal Tab – following character is interpreted as a count of the number of columns by which the current character position is to be displaced from the page left margin. The line (row) position remains unchanged.
0A	LF	Line Feed – couse current line position to be increased by one.
0B	VT	Vertical Tab – following character is interpreted as a count of the number of lines by which the current line (row) position is to be displaced from the page top margin. The position (column) remains unchanged.
0C	FF	Form Feed – resets the character position to the page left and top margins.
0D	NL	New Line – resets character position to the page left margin and increases current line position by one.
86	RLF	Reverse Line Feed – decreases current character line position by one.
8E	HLF	Half Line Feed – increases current character line position by 1/2 line.
8F	HRL	Half Reverse Line Feed – decreases current character line position by 1/2 line.
99	SUP	Superscript – decreases current line postion by 1/4 line and causes character to be drawn at 2/3 font size.
9A	SUB	Subscript – increases current line position by 1/4 line and causes character to be drawn at 2/3 font size.
9B	RSZ	Reduce Size – reduces character to 2/3 font size; column spacing is unchanged.
9C	TRM	Terminate – causes the character string to terminate and interpret next word as a new instruction.
9E	RLC	Restore Line Coordinates – returns to original format by negating effects of SUB, SUP and RSZ.
9F	CUR	Blinking Cursor – causes blinking cursor symbol ⌐ to be drawn at current character position.
7F	DEL	Delete – causes delete symbol ⊠ to be drawn at current character position.

GPU REGISTERS

HEX	DEC	OCT	REGISTER	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	USAGE
07	07	07	CMD.	NPIC	GO	PIC HE	SAM FRM	CPU STK	DEVICES STA#		RBU/DCU STA#		DCU IN	BUFF MODE		CLP TXT	CLP VEC			COMMAND
08	08	10	CTL.	ARB HE	BRK HE		SPIC HE	ELM HE	WRD HE	EDT HE	ERR HE	HIT HE	SEL HE		REG E	SEL E	HIT E		HINT E	CONTROL
09	09	11	STAT.	EDIT NSRT												STATE CODE				STATUS
00	00	00	DIR.	DIRECTORY ADDRESS																
01	01	01	STB.	CPU STACK EXTENSION (LOW ADDRESS)																
02	02	02	SLM.	CPU STACK EXTENSION (HI ADDRESS +1)																
03	03	03	OBA.	CURRENT OBJECT ADDRESS																
04	04	04	IA.	CURRENT INSTRUCTION ADDRESS																ADDRESS RELATED REGISTERS
0A	10	12	PBO.	OBJ# (DIR INDEX) OF BASE PICTURE																
0B	11	13	IR.	CURRENT INSTRUCTION IMAGE																
0C	12	14	OBN.	CURRENT OBJECT NUMBER																
0D	13	15	STK.	CURRENT STACK TOP INDEX																
0E	14	16	SA.	CURRENT STACK LEVEL INDEX																
0F	15	17	GP1.	GENERAL PURPOSE REGISTER #1																
10	16	20	GP2.	GENERAL PURPOSE REGISTER #2																GP REGISTERS
11	17	21	GP3.	GENERAL PURPOSE REGISTER #3																
12	18	22	GP4.	GENERAL PURPOSE REGISTER #4																
13	19	23	PWC.	PICTURE WORD COUNT																
14	20	24	PS.	0	PICTURE SCALE															
15	21	25	PSI.	0	PICTURE INTENSITY DEPTH CUEING															
16	22	26	PDX.	±	PICTURE X DISPLACEMENT															
17	23	27	PDY.	±	PICTURE Y DISPLACEMENT															
18	24	30	PDZ.	0	WINDOW PERSPECTIVE DEPTH CUEING (Z VIEWPOINT)$^{-1}$															PICTURE RELATED REGISTERS
19	25	31	PDI.	0	MAXIMUM PICTURE INTENSITY															
1A	26	32	WCX.	±	WINDOW X CENTER POINT															
1B	27	33	WCY.	±	WINDOW Y CENTER POINT															
1C	28	34	WNZ.	±	3D WINDOW Z NEAR CUTOFF PLANE															
1D	29	35	WSX.	0	WINDOW HORIZONTAL SIZE															
1E	30	36	WSY.	0	WINDOW VERTICAL SIZE															
1F	31	37	WSZ.	0	3D WINDOW Z DEPTH															
20	32	40	DS.	0	INPUT DATA SCALE															

GPU REGISTERS (Continued)

HEX	DEC	OCT	REGISTER	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	USAGE
21	33	41	OS.	0	OBJECT SCALE															OBJECT RELATED REGISTERS
22	34	42	ODX.	±	OBJECT X DISPLACEMENT															
23	35	43	ODY.	±	OBJECT Y DISPLACEMENT															
24	36	44	ODZ.	±	OBJECT Z DISPLACEMENT															
25	37	45	RX.	±	OBJECT X ROTATION															
26	38	46	RY.	±	OBJECT Y ROTATION															
27	39	47	RZ.	±	OBJECT Z ROTATION															
05	05	05	DA.	CURRENT DATA ADDRESS																DATA TABLE REGISTERS
28	40	50	ELN.	ELEMENT NUMBER																
29	41	51	LNCT.				3DI	BLINK		COLOR		LINE TYPE		VECTOR MODE						
2A	42	52	TXCT.				BLINK		COLOR		ORIENTATION		FONT		SIZE					
2B	43	53	COLR.	COLOR																
2C	44	54	PGT.	±	PAGE TOP Y COORDINATE															
2D	45	55	PGL.	±	PAGE LEFT X COORDINATE															
2E	46	56	X.	±	CURRENT X COORDINATE															ELEMENT RELATED REGISTERS
2F	47	57	Y.	±	CURRENT Y COORDINATE															
30	48	60	Z.	±	CURRENT Z COORDINATE															
31	49	61	INTN.	±	CURRENT INTENSITY LEVEL															
32	50	62	DLTX.	±	X INCREMENT															
33	51	63	DLTY.	±	Y INCREMENT															
34	52	64	DLTZ.	±	Z INCREMENT															
35	53	65	DLTI.	±	INTENSITY INCREMENT															
36	54	66	SELWC.	SELECT PICTURE WORD COUNT																SELECT REGISTERS
37	55	67	SELCT.	SEL MODE															ODD BYTE	
38	56	70	HITDEV.	SEL/HIT DEVICE																
39	57	71	PIKX.	±	PICK WINDOW X CENTER															PICK OPTION REGISTERS
3A	58	72	PIKY.	±	PICK WINDOW Y CENTER															
3B	59	73	PIKS.	0	PICK WINDOW SIZE (OPTIONALLY X ONLY)															
3C	60	74	PIKSY.	0	PICK WINDOW Y SIZE (OPTIONAL)															
3D	61	75	HN.											HIT COUNT						HIT REGISTERS
3E	62	76	HITCT.	HIT MODE		DET MODE				HIT UNIT										
3F	63	77	HITWC.	HIT WORD COUNT																
40	64	100	HITEC.	HIT ELEMENT COUNT																
06	06	06	EA.	EDIT ADDRESS																EDIT REGISTERS
41	65	101	EPWC.	EDIT PICTURE WORD COUNT																
42	66	102	ESC.	EDIT SKIP COUNT																
43	67	103	EIC.	EDIT INSERT COUNT																
7B	123	173	LOUT. *											ZL	ZH	YL	YH	XL	XH	
FF	255	377	RWC.	RBU STORE INDEX																

*CLIPPER VIOLATION FLAGS

GPU REGISTER FIELDS

Register	Field	Octal Code	Hex Code	Description
	NPIC	100000	8000	New picture
	GO	40000	4000	Execute/suspend GPU processing
	PIC HE	20000	2000	Picture halt enable
	SAM FRM	10000	1000	Same frame
	DEVICES STA#	00000	0000	Select Devices STA#1
		1000	0200	Select Devices STA#2
		2000	0400	Select Devices STA#3
		3000	0600	Select Devices STA#4
	RBU/DCU STA#	00000	0000	Select RBU/DCU pair #1
		0200	0080	Select RBU/DCU pair #2
CMD.		0400	0100	Select RBU/DCU pair #3
		0600	0180	Select RBU/DCU pair #4
	DCU IN	0100	0040	Initialize DCU
	BUFF MODE	00000	0000	SBM, Single buffer mode
		0020	0010	DBM, Double buffer mode
		0040	0020	E1BM, Edit insert only – buffer mode
		0060	0030	E2BM, Edit insert to end of frame
	CLP TXT	0010	0008	Enable clipping in text mode
	CLP VEC	0004	0004	Enable clipping in vector mode
	BRK HE	40000	4000	Breaklist halt enable
	SPIC HE	10000	1000	Sub-picture halt enable
	ELM HE	4000	0800	Element halt enable
	WRD HE	2000	0400	Word halt enable
	EDT HE	1000	0200	Edit address halt enable
	ERR HE	0400	0100	Error halt enable
CTL.	HIT HE	0200	0080	Hit halt enable
	SEL HE	0100	0040	Select halt enable
	SEL E	0010	0008	Select enable
	HIT E	0004	0004	Hit enable
	EDT E	0002	0002	Edit enable
	HINT E	0001	0001	Interrupt-on-halt enable
	EDIT NSRT	100000	8000	Edit insert
	STATE CODE	00000	0000	Host CPU bus reset
		0001	0001	Ready
		0002	0002	Running
		0003	0003	CPU memory addressing error
STAT.		0004	0004	Invalid graphic instruction
		0005	0005	Invalid argument
	Continued	0006	0006	Illegal register number

GPU REGISTER FIELDS (Continued)

Register	Field	Octal Code	Hex Code	Mnemonic	Description
STAT.	STATE CODE	0007	0007		Graphic stack overflow
		0010	0008		RBU overflow
		0011	0009		Transform out of range
		0012	000A		Invalid PBO or directory structure
		0013	000B		Edit address halt
		0014	000C		Hit halt
		0015	000D		Select halt
		0016	000E		Halt instruction
		0017	000F		Frame halt
		0020	0010		Sub-picture halt
		0021	0011		Breaklist instruction halt
		0022	0012		Element halt
		0023	0013		Word halt
	3DI	10000	1000		Force 3d incremental packing of output
LNCT.	BLINK	0000	0000		No change
		2000	0400		No change
		4000	0800		Disable blink mode
		6000	0C00		Enable blink mode
	COLOR				
	LINE TYPE	0000	0000		No change
		0020	0010		Solid lines
		0040	0020		Long dashes
		0060	0030		Short dashes
		0100	0040		Long/short dashes
		0120	0050		Long/short/short dashes
		0140	0060		Point mode
		0160	0070		Not used
	VECTOR MODE	0010	0008		Curve vector endpoints if incremental
		0004	0004		Place HIT/SELECT in slow mode
TXCT.	BLINK	0000	0000		No change
		2000	0400		No change
		4000	0800		Disable blink mode
		6000	0C00		Enable blink mode
	COLOR				
	ORIENTATION	0000	0000		No change
		0020	0010		No change
		0040	0020		No change
		0060	0030		No change
		0100	0040		Normal (no rotation)
	Continued	0120	0050		90° counterclockwise rotation

GPU REGISTER FIELDS (Continued)

Register	Field	Octol Code	Hex Code	Mnemonic	Description
TXCT.	ORIENTATION	0140	0060		180° rotation
		0160	0070		270° counterclockwise rotation
	FONT	0000	0000		No change
		0004	0004		No change
		0010	0008		Normal font (unslanted)
		0014	000C		Slanted font (right-slanted 26°)
	SIZE	0000	0000		SIZE #1; 120 columns by 60 lines
		0001	0001		SIZE #2; 80 columns by 40 lines (default size)
		0002	0002		SIZE #3; 60 columns by 30 lines
		0003	0003		SIZE #4; 30 columns by 15 lines
SELCT.	SEL MODE	0000	0000		No change in highlighting
		40000	4000		Disable highlighting
		100000	8000		Brighten
		140000	C000		Blink and brighten
	ODD BYTE	0001	0001		Odd byte extension of SELWC
HITDEV	SELREG	0470	0138		Select (pick/pen) device #1 hits for highlighting or HITCT loads
		0500	0140		Select (pick/pen) device #2 hits for highlighting or HITCT loads
		0510	0148		Select (pick/pen) device #3 hits for highlighting or HITCT loads
		0520	0150		Select (pick/pen) device #4 hits for highlighting or HITCT loads
HN.	COUNT	0000	0000		No change
		0001	0001		Generate HIT on 1st "detect"
		0002	0002		Generate HIT on 2nd "detect"
					Generate HIT on "n" "detect"
		0017	000F		Generate HIT on 15th "detect"
HITCT.	HIT MODE	0000	0000		No change
		100000	8000		Disable HIT device HITDEV (pick/pen)
		120000	A000		Enable HIT device HITDEV (PEN: any SW position)
		140000	C000		Disable HIT device HITDEV (PEN: single HIT if SW is ON)
		160000	E000		Enable HIT device HITDEV (PEN: continuous HIT's if SW is ON)
	DET MODE	0000	0000		No change
		4000	0800		No change
		10000	1000		Disable hits for HITDEV
		14000	1800		Enable HN counted hits for HITDEV
	HIT UNIT	0200	0080		(pick/pen) device #1 generating hit (valid during Hit Halt)
		0100	0040		(pick/pen) device #2 generating hit (valid during Hit Halt)
		0040	0020		(pick/pen) device #3 generating hit (valid during Hit Halt)
		0020	0010		(pick/pen) device #4 generating hit (valid during Hit Halt)

DISPLAY SYSTEM ADDRESS ORGANIZATION

GPU/STA #	UNIT	GP BUS ADDR		INT ID	COMMENTS
		HEX	OCT	OCT	
GPU #1	GPU REGISTERS	000-07F	0000-0177	02	SEE APPENDICES B1, B2
	WCS #1	340	1500		LOAD MEMORY ADDRESS
	WCS #1	341	1501		I/O TOP 16
	WCS #1	342	1502		I/O BOTTOM 8
	WCS #1	343	1503		I/O CONSTANT
GPU #2	GPU REGISTERS	080-0FF	0200-0377	03	SEE APPENDICES B1, B2
	WCS #2	344	1504		LOAD MEMORY ADDRESS
	WCS #2	345	1505		I/O TOP 16
	WCS #2	346	1506		I/O BOTTOM 8
	WCS #2	347	1507		I/O CONSTANT
STA #1	DCU #1/DGU's	100-17F	0400-0577	04	
	RBU #1	300-30F	1400-1417		
	DATA TABLET	380-382	1600-1602	40	
	FUNCTION SWITCHES	384-386	1604-1606	44	
	KEYBOARD	387	1607	47	
	JOY STICK	393-395	1623-1625		
	CONTROL DIALS	396-39F	1626-1637		
STA #2	DCU #2/DGU's	180-1FF	0600-0777	05	
	RBU #2	310-31F	1420-1437		
	DATA TABLET	3A0-3A2	1640-1642	50	
	FUNCTION SWITCHES	3A4-3A6	1644-1646	54	
	KEYBOARD	3A7	1647	57	
	JOY STICK	3B3-3B5	1663-1665		
	CONTROL DIALS	3B6-3BF	1666-1677		
STA #3	DCU #3/DGU's	200-27F	1000-1177	06	
	RBU #3	320-32F	1440-1457		
	DATA TABLET	3C0-3C2	1700-1702	60	
	FUNCTION SWITCHES	3C4-3C6	1704-1706	64	
	KEYBOARD	3C7	1707	67	
	JOY STICK	3D3-3D5	1723-1725		
	CONTROL DIALS	3D6-3DF	1726-1737		
STA #4	DCU #4/DGU's	280-2FF	1200-1377	07	
	RBU #4	330-33F	1460-1477		
	DATA TABLET	3E0-3E2	1740-1742	70	
	FUNCTION SWITCHES	3E4-3E6	1744-1746	74	
	KEYBOARD	3E7	1747	77	
	JOY STICK	3F3-3F5	1763-1765		
	CONTROL DIALS	3F6-3FF	1766-1777		

HARDWARE AND DEVICE REGISTERS

ACCESS	HEX	OCT	REGISTER	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
PO,RFRSH	100	0400	DCU.	START/STOP			EN SET	CLK	DEV*	EOL	HLT	ACK	FRAME MODE			REFRESH RATE			
PO,RFRSH	101	0401	SEL.	SEL MODE		SELECT REGISTER				HIT SELECT REGISTER				SLOW					
PI	102	0402	DCUST.				RUN	2nd BYTE	CLK	DEV*	EOL	HLT	HIT1	HIT2	HIT3	HIT4	PSW1	PSW2	PSW3	PSW4
PI	104	0404	DCUWC.	RBU WORD COUNT															
PI,RFRSH	105	0405	DCUI.	OP CODE		FIELD										1		0	
PO,RFRSH	108	0410	SELR1.	RBUWC (GPU) OR SAVED RBU WORD COUNT (DCU)															
PO,RFRSH	109	0411	SELR2.	RBUWC (GPU) OR SAVED RBU WORD COUNT (DCU)															
PO,RFRSH	10A	0412	SELR3.	RBUWC (GPU) OR SAVED RBU WORD COUNT (DCU)															
PO,RFRSH	10B	0413	SELR4.	RBUWC (GPU) OR SAVED RBU WORD COUNT (DCU)															
PIO,RFRSH	10C	0414	SPX.	±	CHARACTER SPACE VALUE														
PIO,RFRSH	10D	0415	SPY.	±	CHARACTER SPACE VALUE														
PIO,RFRSH	10E	0416	SPZ.	±	CHARACTER SPACE VALUE														
PIO,RFRSH	10F	0417	NAME.																
PI	110	0420	HIT1.	SAVED RBU WORD COUNT															
PI	111	0421	HIT2.	SAVED RBU WORD COUNT															
PI	112	0422	HIT3.	SAVED RBU WORD COUNT															
PI	113	0423	HIT4.	SAVED RBU WORD COUNT															
PIO,RFRSH	114	0424	ATRC.																
PIO,RFRSH	115	0425	ATRX.																
PIO,RFRSH	116	0426	ATRY.																
PO,RFRSH	120	0440	VGU.										LINE TYPE						
PIO,RFRSH	124	0444	XREG.	±	COORDINATE													OF	
PIO,RFRSH	125	0445	YREG.	±	COORDINATE													OF	
PIO,RFRSH	126	0446	ZREG.	±	COORDINATE													OF	
PIO,RFRSH	127	0447	IREG.	VALUE															
PO,RFRSH	128	0450	MCU.	EN MS								MS1	MS2	MS3	MS4	MS5	MS6	MS7	MS8
PO,RFRSH	129	0451	VEL.			VALUE													
PO,RFRSH	12A	0452	COLR.							VALUE									
PIO,RFRSH	12B	0453	CCTL.	EN DCU CLR	DIS CLR VEL				VALUE										
PO,RFRSH	130	0460	FGU.											ROTATION		SLANT			
PO,RFRSH	131	0461	CHAR.											CHARACTER CODE					
PO,RFRSH	132	0462	RAMS.					STARTING ADDRESS											
PO,RFRSH	133	0463	RAML.	RAM LOAD DATA															
PO,RFRSH	134	0464	CHRX.	±	CR11 VALUE					±		CR12 VALUE							
PO,RFRSH	135	0465	CHRY.	±	CR21 VALUE					±		CR22 VALUE							
PO,RFRSH	137	0467	CHSC.	±	CHARACTER SCALE														
PO,RFRSH	138	0470	HITC1.	EN HIT		DET MODE										DET COUNT			
PO,RFRSH	139	0471	PIKX1.	±	X CENTER POSITION														
PO,RFRSH	13A	0472	PIKY1.	±	Y CENTER POSITION														
PO,RFRSH	13B	0473	PIKS1.	WINDOW SIZE (OPTIONALLY X ONLY)															
PO,RFRSH	13C	0474	PKSY1.	WINDOW Y SIZE (OPTIONAL)															

* Pick window or light pen only.

NOTE: PI AND PO ARE GP BUS DATA TRANSFERS BETWEEN GPU AND DCU.
RFRSH IS THE REFRESH DISPLAY LIST FROM THE RBU.

HARDWARE AND DEVICE REGISTERS (Continued)

ACCESS	ADDRESS HEX	OCT	REGISTER	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
PO,RFRSH	140	0500	HITC2.	EN HIT			DET MODE									DET COUNT			
PO,RFRSH	141	0501	PIKX2.	±	X CENTER POINT														
PO,RFRSH	142	0502	PIKY2.	±	Y CENTER POINT														
PO,RFRSH	143	0503	PIKS2.	WINDOW SIZE (OPTIONALLY X ONLY)															
PO,RFRSH	144	0504	PKSY2.	WINDOW Y SIZE (OPTIONAL)															
PO,RFRSH	148	0510	HITC3.	EN HIT			DET MODE									DET COUNT			
PO,RFRSH	149	0511	PIKX3.	±	X CENTER POSITION														
PO,RFRSH	14A	0512	PIKY3.	±	Y CENTER POSITION														
PO,RFRSH	14B	0513	PIKS3.	WINDOW SIZE (OPTIONALLY X ONLY)															
PO,RFRSH	14C	0514	PKSY3.	WINDOW Y SIZE (OPTIONAL)															
PO,RFRSH	150	0520	HITC4.	EN HIT			DET MODE									DET COUNT			
PO,RFRSH	151	0521	PIKX4.	±	X CENTER POINT														
PO,RFRSH	152	0522	PIKY4.	±	Y CENTER POINT														
PO,RFRSH	153	0523	PIKS4.	WINDOW SIZE (OPTIONALLY X ONLY)															
PO,RFRSH	154	0524	PKSY4.	WINDOW Y SIZE (OPTIONAL)															
PI	300	1400	RBUST.	IH1	EM1	EM2	EDT	EIP	E1E	E2E	END	EON				SIZE			
PI	301	1401	RBUB.	BUF 1 BASE ADDRESS															
PIO	302	1402	RBUP1.	BUF 1 POINTER															
PIO	303	1403	RBUP2.	BUF 2 BASE ADDRESS															
PIO	304	1404	RBU1E.	BUF 1 EDIT BASE ADDRESS															
PIO	305	1405	RBU2E.	BUF 2 EDIT BASE ADDRESS															
PIO	306	1406	RBUMAR.	MEMORY ADDRESS															
PIO	307	1407	RBUDAT.	RBU DATA															
USER	380	1600	DTX.1	±															
USER	381	1601	DTY.1	±															
USER	382	1602	DTS.1													XOS	YOS	PNN	PRS	IEN
USER	384	1604	FSL0.1	S00-15 (READ), L00-15 (WRITE)															
USER	385	1605	FSL1.1	S16-31 (READ), L16-31 (WRITE)															
USER	386	1606	FSKC.1				SD1		SD0							LD1	IE1	LD0	IE0
USER	387	1607	KB.1	KDV	KIE									DATA					
USER	393	1623	JSX.1	±															
USER	394	1624	JSY.1	±															
USER	395	1625	JSZ.1	±															
USER	396	1626	DL1.1 (1)	±															
USER	3A0	1640	DTX.2	±															
USER	3A1	1641	DTY.2	±															
USER	3A2	1642	DTS.2													XOS	YOS	PNN	PRS	IEN

HARDWARE AND DEVICE REGISTERS (Continued)

ACCESS	ADDRESS HEX	OCT	REGISTER	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
USER	3A4	1644	FSL0.2	S00–15 (READ), L00–15 (WRITE)														
USER	3A5	1645	FSL1.2	S16–31 (READ), L16–31 (WRITE)														
USER	3A6	1646	FSKC.2					SD1		SD0						LD1	IE1	LD0	IE0
USER	3A7	1647	KB.2	KDV	KIE								DATA						
USER	3B3	1663	JSX.2	±															
USER	3B4	1664	JSY.2	±															
USER	3B5	1665	JSZ.2	±															
USER	3B6	1666	DL1.2 (2)	±															
USER	3C0	1700	DTX.3	±															
USER	3C1	1701	DTY.3	±															
USER	3C2	1702	DTS.3													XOS	YOS	PNN	PRS	IEN
USER	3C4	1704	FSL0.3	S00–15 (READ), L00–15 (WRITE)														
USER	3C5	1705	FSL1.3	S16–31 (READ), L16–31 (WRITE)														
USER	3C6	1706	FSKC.3					SD1		SD0						LD1	IE1	LD0	IE0
USER	3C7	1707	KB.3	KDV	KIE								DATA						
USER	3D3	1723	JSX.3	±															
USER	3D4	1724	JSY.3	±															
USER	3D5	1725	JSZ.3	±															
USER	3D6	1726	DL1.3 (3)	±															
USER	3E0	1740	DTX.4	±															
USER	3E1	1741	DTY.4	±															
USER	3E2	1742	DTS.4													XOS	YOS	PNN	PRS	IEN
USER	3E4	1744	FSL0.4	S00–15 (READ), L00–15 (WRITE)														
USER	3E5	1745	FSL1.4	S16–31 (READ), L16–31 (WRITE)														
USER	3E6	1746	FSKC.4					SD1		SD0						LD1	IE1	LD0	IE0
USER	3E7	1747	KB.4	KDV	KIE								DATA						
USER	3F3	1763	JSX.4	±															
USER	3F4	1764	JSY.4	±															
USER	3F5	1765	JSZ.4	±															
USER	3F6	1766	DL1.4 (4)	±															

(1) DIALS 2 THROUGH 10 OF CD#1 USE OCTAL ADDRESSES 1627 THROUGH 1637; MNEMONICS DL2.1 TO DL10.1.

(2) DIALS 2 THROUGH 10 OF CD#2 USE OCTAL ADDRESSES 1667 THROUGH 1677; MNEMONICS DL2.2 TO DL10.2.

(3) DIALS 2 THROUGH 10 OF CD#3 USE OCTAL ADDRESSES 1727 THROUGH 1737; MNEMONICS DL2.3 TO DL10.3.

(4) DIALS 2 THROUGH 10 OF CD#4 USE OCTAL ADDRESSES 1767 THROUGH 1777; MNEMONICS DL2.4 TO DL10.4.

STANDARD ASCII CHARACTER SET
(Reference Appendix A5 for User Control Codes)

MSD \ LSD	∅	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
∅									BS	HT	LF	VT	FF	NL		
1																
2	Space	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	⊠
8							RLF								HLF	HRL
9										SUP	SUB	RSZ	FSX	GSX	RLC	CUR

NOTE: CODES 21 THROUGH 7F AND 9F ARE PRINTABLE CHARACTERS.

HARDWARE DECODING OF CONTROL CODES

Mnemonic	Decoded By	Action Taken
BS	DCU	Subtract SPACE X, Y, Z coordinates from current X, Y, Z coordinates and send result to VGU along with the GO bit.
HT LF VT FF NL RLF HLF HRL	GPU	Generate a GSX control code followed by vector endpoint coordinates for a move to the next character position. (Continued on next page)

HARDWARE DECODING OF CONTROL CODES (Continued)

Mnemonic	Decoded By	Action Taken
GSX	DCU	The DCU decodes this control code and interprets the next word as position data (same as vector absolute data format) to signal the VGU to perform a vector move (or draw) to a new character position. The OF field of 01 or 11 (load and move or load and draw) in the final position data word then returns the DCU to the character string. In addition to the above (and prior to a VGU "move" or "draw", the SCALE register in the FGU may be updated by specifying a REG field of 11 in the data word.
FSX	DCU	The DCU decodes this control code to terminate the character string and interpret the following word as a new instruction.
SUB	FGU	The FGU decodes this control code to perform the subscript function (1/4 line feed, 2/3 font size). The subscript function remains active until control code RLC is received.
SUP	FGU	The FGU decodes this control code to perform the superscript function (1/4 reverse line feed, 2/3 font size). The superscript function remains active until control code RLC is received.
RSZ	FGU	The FGU decodes this control code to reduce character size (2/3 font size). The size reduction remains active until control code RLC is received.
RLC	FGU	The FGU decodes this control code to negate the SUB, SUP, and RSZ functions.
Space	FGU	This code forces the FGU to remain in the "parking-position" of the character space and signals the VGU to space to the next character position.
CUR	DCU/FGU	This code is decoded by the DCU to enter the character blink mode (4-Hz). The code is also transferred to the FGU where it is decoded to perform the cursor draw. The VGU performs no character spacing after the draw.

INDEX

INDEX (Continued)